UNIVERSIDAD POLITÉCNICA DE MADRID

Facultad de Informática

Departamento de Inteligencia Artificial

TESIS DOCTORAL

Arquitectura cognitiva basada en el gradiente sensorial y su aplicación a la Robótica Móvil

Francisco Serradilla García

Directores: Darío Maravall Gómez-Allende

José Gabriel Zato Recellado

Madrid, 1997

Resumen

El objetivo de esta tesis es el desarrollo de un sistema completo de navegación, aprendizaje y planificación para un robot móvil. Dentro de los innumerables problemas que este gran objetivo plantea, hemos dedicado especial atención al problema del *conocimiento autónomo* del mundo.

Nuestra mayor preocupación ha sido la de establecer mecanismos que permitan, a partir de información sensorial *cruda*, el desarrollo incremental de un modelo topológico del entorno en el que se mueve el robot. Estos mecanismos se apoyan invariablemente en un nuevo concepto propuesto en esta tesis: *el gradiente sensorial*. El gradiente sensorial es un dispositivo matemático que funciona como un detector de sucesos interesantes para el sistema. Una vez detectado uno de estos sucesos, el robot puede identificar su situación en un mapa topológico y actuar en consecuencia. Hemos denominado a estas situaciones especiales *lugares sensorialmente relevantes*, ya que (a) captan la atención del sistema y (b) pueden ser identificadas utilizando la información sensorial.

Para explotar convenientemente los modelos construidos, hemos desarrollado un algoritmo capaz de elaborar *planes internalizados*, estableciendo una red de sugerencias en los lugares sensorialmente relevantes, de modo que el robot encuentra en estos puntos una dirección recomendada de navegación.

Finalmente, hemos implementado un sistema de navegación robusto con habilidades para interpretar y adecuar los planes internalizados a las circunstancias concretas del momento. Nuestro sistema de navegación está basado en la teoría de campos de potencial artificial, a la que hemos incorporado la posibilidad de añadir cargas ficticias como ayuda a la evitación de mínimos locales.

Como aportación adicional de esta tesis al campo genérico de la ciencia cognitiva, todos estos elementos se integran en una arquitectura centrada en la memoria, lo que pretende resaltar la importancia de ésta en los procesos cognitivos de los seres vivos y aporta un giro conceptual al punto de vista tradicional, centrado en los procesos.

Summary

The general objective of this thesis is the development of a global navigation system endowed with planning and learning features for a mobile robot. Within this general objective we have devoted a special effort to the *autonomous learning* problem.

Our main concern has been to establish the necessary mechanisms for the incremental development of a topological model of the robot's environment using the sensory information. These mechanisms are based on a new concept proposed in the thesis: the *sensory gradient*. The sensory gradient is a mathematical device which works like a detector of "interesting" environment's events. Once a particular event has been detected the robot can identify its situation in the topological map and to react accordingly. We have called these special situations *relevant sensory places* because (a) they capture the system's attention and (b) they can be identified using the sensory information.

To conveniently exploit the built-in models we have developed an algorithm able to make *internalized plans*, establishing a suggestion network in the sensory relevant places in such way that the robot can find at those places a recommended navigation direction.

It has been also developed a robust navigation system able to navigate by means of interpreting and adapting the internalized plans to the concrete circumstances at each instant, i.e. a reactive navigation system. This reactive system is based on the artificial potential field approach with the additional feature introduced in the thesis of what we call *fictitious charges* as an aid to avoid local minima.

As a general contribution of the thesis to the cognitive science field all the above described elements are integrated in a memory-based architecture, emphasizing the important role played by the memory in the cognitive processes of living beings and giving a conceptual turn in the usual process-based approach.

El magrebí le dijo: "¡Aladino! Fíjate y haz exactamente todo lo que te voy a decir; no te olvides de nada. Baja con mucho cuidado al fondo del subterráneo; una vez abajo encontrarás un lugar dividido en cuatro partes: en cada una de ellas verás cuatro jarrones de oro y otros objetos de oro y plata; no los toques, no cojas nada de ellos; sigue adelante hasta llegar al cuarto compartimento, y procura que tu ropa no toque los jarrones ni las paredes; no te detengas ni un momento, pues si lo hicieras, inmediatamente te metamorfosearías y te transformarías en una piedra negra. Al llegar al cuarto compartimento verás una puerta: ábrela, y pronuncia los nombres que has dicho al levantar la losa. Entra: te encontrarás en un jardín, adornado con árboles y frutos. Avanza cincuenta codos por el camino que tengas delante: llegarás a un salón, del cual arranca una escalera de unos treinta peldaños. Fíjate en el techo, verás que de él cuelga una lampara."

De *Las mil y una Noches* (noche 525a en la edición de Juan Vernet)

Je dirai que j'ai trouvé la demonstration de tel théorème dans telles circounstances;
ce théorème aura un nom barbare, que beaucoup d'entre vous ne conaîtront pas;
mais cela n'a pas d'importance: ce qui est interessant pour le psychologue,
ce n'est pas le théorème, ce sont les circonstances.

Henri Poincaré, *L'invention Mathématique*:

Agradecimientos

A Zato y a Darío, directores de esta tesis; a Zato por haber sido el principal responsable de mi formación a lo largo de tantos años, y a Darío por su apoyo durante el desarrollo de este trabajo, y también por iniciar en la Facultad de Informática una línea tan interesante como es la robótica móvil.

A Javi, por un millar de motivos, entre los que citaré la revisión de los textos, las múltiples discusiones científicas de las mañanas de los ¿miércoles?, los fines de semana pasados en los pasillos de la Facultad, su labor como mecanismo de seguridad adicional del robot, su actuación como director de fotografía de la película y, por último, pero no menos importante, por cargar *con* las baterías y salir indemne.

A María José, por desenredar mis enmarañados planteamientos en relación con la actualización bayesiana de arcos.

A Luis, por ayudarme, junto con Ana García Serrano, a confeccionar el DFD de los trámites administrativos previos a la defensa de la tesis.

A todos mis compañeros en la EUI, por el tiempo robado a otros temas, sin ir más lejos al de encontrar dónde diantres está cortada la red. A mis alumnos de TFC, por prescindir, como mínimo, de nueve décimas partes de director.

A Pulgarcito (el robot), por asumir con estoicismo el miedo a la colisión durante la depuración del código.

A Juan, a Alexis, y de nuevo a María José, por las cartas que he dejado de escribirles, por las discusiones que hemos dejado de tener, por los libros que hemos dejado de escribir.

Índice

1. Introducción

La Robótica y la Inteligencia Artificial (IA) son campos fascinantes. El desarrollo de mecanismos que posean cierto grado de movilidad y autonomía para interactuar con el mundo real es una cuestión a la que los científicos han dedicado un gran esfuerzo en los últimos años, y los visionarios tanto o más que éstos en años anteriores. Pero la construcción de sistemas robóticos inteligentes capaces de desenvolverse en entornos desconocidos constituye un profundo desafío. Los sistemas deben ser autosuficientes, robustos y equipados para responder oportunamente a sucesos inesperados. Nacida con esta finalidad, la Robótica Móvil es una nueva disciplina que aúna los problemas de la Robótica y la IA, junto con toda una serie de nuevos problemas específicos.

Los primeros proyectos de robótica móvil datan de principios de los años 70, cuando por primera vez se reúnen en una misma plataforma sensores, motores y procesadores. A partir de entonces, como veremos, se han sucedido diversos modos de entender el problema y, en consecuencia, diversas maneras de afrontarlo.

En este capítulo intentaremos dar una visión general de los objetivos de la robótica móvil como campo de estudio, justificando el interés de la investigación en el área y analizando los enfoques más importantes. Posteriormente se analizarán las posibles relaciones entre los problemas planteados en el diseño de sistemas autónomos para robótica móvil con una serie de tópicos estudiados por la psicología cognitiva.

1.1 El problema de la robótica móvil

El problema central del que se ocupa la investigación actual en robótica móvil es el de diseñar y construir sistemas capaces de desenvolverse de modo autónomo en el mundo real, capturando información sensorial y utilizándola para alcanzar objetivos determinados. Esta información puede servir, al mismo tiempo, para construir y refinar un modelo del entorno del robot que ayude a planificar misiones posteriores. Esta definición incide fundamentalmente en dos aspectos: (a) el mundo no es conocido de antemano ni puede serlo completamente —entre otras cosas porque está sujeto a cambios permanentes producidos por otros agentes al margen del robot— lo que conduce a (b) la necesidad de una total autonomía en los sistemas.

Esta ambiciosa tarea involucra una enorme cantidad de subtareas. Atendiendo a la división habitual realizada en la mayoría de las recopilaciones de artículos y actas de congresos [Iyengar & Elfes, 1991a; 1991b; Charnley, 1993; Halme & Koskinen, 1995; Cox & Wilfong, 1990] podemos identificar como problemas más relevantes los de *percepción*, *modelado del mundo*, *planificación de caminos*, *navegación*, *control*, *planificación de tareas* y *toma de decisiones*. La presente tesis se ocupa en mayor o menor medida de todos ellos, aunque poniendo especial énfasis en los tres primeros.

1.2 ¿Por qué robótica móvil?

Las aplicaciones potenciales de los robots móviles son innumerables [Iyengar & Elfes, 1991a; 1991b; Cox & Wilfong, 1990; Kanade *et al*, 1994; Hollier, 1987], desde la operación en entornos industriales u hostiles (centrales nucleares, minas, profundidades oceánicas) hasta la realización de tareas domésticas (limpieza, vigilancia), pasando por el desarrollo de diversos sistemas de apoyo en el sector servicios (robots enfermeras, transporte de pasajeros o mercancías, etc.), las ayudas técnicas a personas discapacitadas o la exploración espacial a distancias en las que la teleoperación no es posible. Pero no son únicamente sus potenciales aplicaciones las que hacen que la robótica móvil se haya convertido en uno de los campos más frecuentados por los investigadores en los últimos tiempos. Hay otra razón, quizá actualmente de mayor peso: el interés teórico del problema.

La robótica móvil es actualmente un riguroso campo de pruebas para la disciplina de la IA. Los científicos, habida cuenta de las limitaciones denunciadas en los *problemas de juguete*[1] [Brooks, 1991], desean probar sus teorías en aplicaciones más cercanas a la realidad. La robótica móvil reúne todos los elementos necesarios:

- Incluye tareas tradicionalmente excluidas de los planteamientos simbólicos de la IA, como son el *problema de la percepción* y el *problema de la actuación*, que en el presente se consideran problemas al menos tan importantes como el razonamiento.
- Requiere enfrentarse a eventualidades no previsibles en un modelo a priori. La teoría del dominio no debe ni puede ser completamente preestablecida.
- En la IA clásica, los modelos del mundo son proporcionados por el diseñador del sistema; pero un sistema verdaderamente autónomo debe construir sus propios modelos. De este modo, las tareas de aprendizaje se hacen imprescindibles de una manera natural, y no como un añadido o una mejora a los sistemas.
- Un sistema no sirve si no funciona en la realidad; es decir, la validez de un planteamiento debe evaluarse analizando su comportamiento en el mundo real. Esto podría considerarse como el análogo actual al *test* de Turing.
- Para que los sistemas funcionen en la realidad, es necesario que su respuesta sea suficientemente rápida. Los modelos computacionalmente costosos no sirven.

En definitiva, la robótica móvil obliga a los investigadores a adoptar métodos prácticos para resolver problemas reales. Estas necesidades prácticas han hecho surgir nuevos planteamientos para la IA, que han provocado una revitalización de la disciplina.

1.3 Distintos paradigmas de diseño de sistemas autónomos

Podemos clasificar los distintos modos de enfrentar el problema de la robótica móvil, en particular, y de los sistemas autónomos, en general, en tres grandes grupos. En primer lugar tenemos el que podríamos denominar “enfoque clásico”, que se extiende en el tiempo desde

1 Un problema de juguete es un problema simple que se espera que capture los aspectos esenciales de los problemas reales. Los problemas de juguete se han usado tradicionalmente para evaluar las capacidades de diferentes algoritmos de IA; un caso típico es el conocido *mundo de los cubos*.

los trabajos de Nilsson [Nilsson, 1969] hasta alrededor de 1986 (aunque numerosos investigadores continúan hoy día situados dentro de este paradigma); en segundo lugar nos referiremos al "enfoque reactivo", que se populariza con los trabajos de Brooks [Brooks, 1986] si bien tiene antecedentes en el terreno de la psicología [Gibson, 1977; Piaget, 1987]; y finalmente existe un "enfoque híbrido" —del que se comienza a hablar alrededor de 1990— cuyos planteamientos intentan sacar partido de los dos enfoques anteriores. En lo que sigue comentaremos brevemente estos tres paradigmas.

1.3.1 Enfoque clásico

El enfoque clásico o formalista [Maravall, 1996] se centra principalmente en el problema de la planificación de trayectorias, suponiendo en general que el modelo del entorno (completo o parcial) se conoce a priori, y utiliza técnicas predominantemente geométricas. Problemas como la incertidumbre en las medidas sensoriales o en la posición, la existencia de otros objetos móviles o las limitaciones cinemáticas se excluyen del planteamiento básico y se consideran como extensiones a los modelos. El desarrollo automático de modelos —cuando existe— se concibe como un proceso aparte, y su resultado es una estructura simbólica que permite obtener planes tentativos en forma de subobjetivos que el robot intentará alcanzar secuencialmente. Los sistemas sensoriales más utilizados son la visión artificial [Kosaka & Kak, 1992] y los ultrasonidos [Leonard & Durrant-Whyte, 1992]. Una extensa descripción del enfoque clásico puede encontrarse en [Latombe, 1993] y en [Meystel, 1991]. Algunos de los primeros prototipos desarrollados se describen en [Maravall *et al*, 1989].

En los enfoques clásicos la planificación se realiza en el llamado espacio de configuración [Lozano-Pérez, 1987]. En éste, el robot se representa como un punto en un espacio n-dimensional (una por cada dimensión del entorno, usualmente 2 o 3), y los obstáculos se corresponden con regiones prohibidas en este espacio. Además, las regiones prohibidas se transforman, haciéndolas crecer de modo que el robot pueda considerarse adimensional a efectos del cálculo de la trayectoria a seguir [Lozano-Pérez & Wesley, 1979]. Obsérvese que el cálculo del crecimiento de regiones requiere en principio un conocimiento completo del entorno.

Las técnicas principales de planificación utilizadas por los enfoques clásicos se pueden resumir en tres grupos:

1) **Mapas de carreteras** (*roadmaps*): intentan capturar la conectividad del espacio libre en una red de curvas unidimensionales. Los más utilizados son:
   a) *Grafo de visibilidad* [Nilsson, 1969]. Es un grafo no dirigido cuyos nodos son las posiciones inicial y final del robot y los vértices de los objetos (representados como figuras poligonales), a los que previamente se les ha aplicado el algoritmo de crecimiento de regiones. Los arcos del grafo incluyen todas las posibles conexiones entre nodos siempre y cuando no intersecten el interior de ninguno de los objetos expandidos. Utilizando el grafo de visibilidad es posible encontrar una ruta libre de colisiones para ir desde la posición inicial a la final.
   b) *Diagramas de Voronoi* [Ó'Dúnlaing & Yap, 1982]. Grafo cuyos arcos están formados por puntos que maximizan la distancia entre el robot y los obstáculos. Los nodos se sitúan en los puntos de intersección de dos o más arcos.
   c) *Regiones libres* [Brooks, 1983]. Se basa en extraer estructuras geométricas (*freeways*) que modelizan las regiones libres y construir con ellas un grafo.
2) **Descomposición en celdas** [Chatila, 1982; Elfes, 1989; 1990; 1991]. Pese a sus inconvenientes, es uno de los más populares en la actualidad. Estos métodos dividen el espacio en regiones simples, llamadas celdas, que forman una retícula (*grid*). Para realizar búsquedas se utiliza la relación de adyacencia entre las celdas, de modo que desde cada celda se puede transitar a una de las ocho celdas colindantes. La gran ventaja que presentan es la simplicidad a la hora de integrar y actualizar información proveniente de múltiples sensores. Su mayor inconveniente es el elevado coste computacional que requiere la planificación de misiones debido a la alta conectividad del modelo y al gran número de celdas necesario para representar el entorno, incluso en modelos de tamaño reducido. Además es necesario utilizar mecanismos de planificación que obtengan trayectorias lineales a partir de una secuencia de celdas [Thorpe, 1984].
3) **Campos de potencial artificial** [Khatib, 1986; Krogh, 1984; Latombe, 1993]. Esta teoría intenta asimilar el problema de un robot moviéndose en un entorno con obstáculos al de una carga eléctrica inmersa en un campo de potencial. Para ello, se considera que la meta genera una fuerza atractiva y los obstáculos generan fuerzas repulsivas, de modo que la

carga que representa al robot intentará aproximarse a la meta eludiendo los obstáculos. Las aplicaciones clásicas de esta teoría suponen que el mundo se conoce a priori; a partir de la descripción del mundo se extrae la posición y la intensidad de las cargas y con todo ello se calcula la trayectoria que deberá seguir el robot. Una alternativa sin duda más interesante —que podríamos considerar ya dentro del enfoque reactivo— es utilizar los sensores del robot para obtener información y reaccionar basándose en estos datos. Por ejemplo, si el robot está equipado con sensores de proximidad, se puede usar un campo de potencial para planificar *on-line* el movimiento del robot. El problema que presenta la teoría de campos de potencial artificial es su elevada sensibilidad a los mínimos locales; escapar del mínimo precisa algún tipo de exploración por ensayo y error. La ventaja de estos métodos es su baja carga computacional, lo que hace posible su utilización en tiempo real.

En cuanto a la arquitectura de los robots, el paradigma clásico se basa en el esquema tradicional basado en las tareas típicas de un sistema inteligente: *percibir, razonar y actuar* [Winston, 1994]. Estas tres tareas se encadenan secuencialmente en un bucle de control que se repite indefinidamente. Según este planteamiento, el bucle de control (a) capta información del entorno, que le ayuda a (b) actualizar sus modelos internos y a planificar la próxima acción y, finalmente, (c) el sistema comunica a los efectores el movimiento a realizar y vuelve a la fase de percepción [Maravall, 1996].

1.3.2 Enfoque reactivo

Quizá debido a que la construcción de modelos es un proceso extremadamente complejo, y el paradigma clásico no proporcionaba soluciones satisfactorias, los partidarios del enfoque reactivo decidieron simplemente obviar la creación de modelos [Brooks, 1986; 1989; 1991] y con ello los procesos de planificación. En palabras del propio Brooks los sistemas reactivos *"usan el mundo como su propia representación"*.

Esta decisión, además de simplificar los sistemas, dio origen a un fructífero paradigma en el que la información suministrada por el entorno sirve de única guía a la toma de decisiones inmediata. Es posible realizar, de manera puramente reactiva, muchas más tareas de las que se pudiera imaginar. El énfasis se pone en la información sensorial de los más variados tipos:

infrarrojos, luz, sonido, sensores de contacto. La visión artificial pasa a situarse en segundo plano —aunque no se abandona—, ya que es un campo que requiere procesos computacionalmente costosos y los investigadores en robótica reactiva buscan ante todo sistemas reales que funcionen en tiempo real.

Un sistema reactivo está compuesto por un conjunto de procesos simples, cada uno con su propio bucle de percepción-razonamiento-acción, que cooperan para alcanzar un objetivo común. Un sistema de *arbitraje* decide quién toma el control del robot en cada instante. La mente de un robot reactivo consiste, según Connell [Connell, 1990, pag. 11], en una *"colección esquizofrénica de impulsos que compiten por el control del cuerpo"*. Es muy interesante la visión de Arkin [Arkin, 1989] en la que se relacionan los comportamientos con los *esquemas sensoriomotrices* de Piaget [Piaget, 1987]. Un comportamiento se genera por la activación de una instancia de esquema, y esta instancia de esquema se activa incitada por la aparición de cierto conjunto de estímulos sensoriales.

El pionero de este enfoque neoconductista es sin duda Rodney Brooks, que en 1986 propone una de las primeras arquitecturas reactivas basada en la idea de subsunción [Brooks, 1986]. Su sistema está compuesto por una serie de procesos jerárquicos donde los de mayor nivel subsumen a los de menor nivel. Esta subsunción puede manifestarse de dos modos: (a) inyectando en el módulo subsumido nueva información que sustituye a la información de entrada, y (b) inhibiendo alguna de las salidas de módulo subsumido.

Para los reactivos la arquitectura y la planificación son dos cuestiones estrechamente relacionadas; ambas están integradas y entremezcladas en una red de comportamientos que depende exclusivamente de los datos sensoriales. El ingeniero diseña la arquitectura, y los comportamientos emergen de ella. La planificación está imbuida en la arquitectura [Kaelbling, 1987] y no se define explícitamente, aunque algunas propuestas recientes [Mataric, 1992] manifiestan tímidos intentos de modelar el mundo y efectuar algún tipo de planificación posterior explícita. Estas propuestas originan un nuevo paradigma: el de los *enfoques híbridos*.

Dentro del paradigma reactivo surge el problema del *arbitraje de los comportamientos*, que consiste en decidir qué agente debe activarse en cada instante. Existen dos enfoques opuestos

a la hora de realizar el arbitraje, según tengamos comportamientos excluyentes (enfoque de Brooks) o comportamientos cooperantes (enfoque de Arkin). En los comportamientos excluyentes sólo un comportamiento tiene el control del robot en cada instante; el sistema de arbitraje decide qué comportamiento toma el control. En los cooperantes, la salida de varios comportamientos simultáneos se suma para obtener la salida global del sistema. En nuestra opinión, ambos enfoques son complementarios, y en nuestras acciones cotidianas pueden encontrarse ejemplos de comportamientos tanto excluyentes como cooperantes.

Normalmente, el esquema de arbitrajes está preestablecido en el diseño del sistema de control del robot y no puede cambiarse. Maes y Brooks [Maes & Brooks, 1990] proponen un interesante método para que el sistema aprenda cuándo debe utilizar cada comportamiento, analizando la correlación entre la activación de un comportamiento y el progreso obtenido en la tarea a desarrollar.

Dos campos muy relacionados con la robótica reactiva son el de la Vida Artificial [Langton, 1989; Varela & Bourgine, 1992; Brooks, 1992], que persigue la construcción de organismos artificiales autónomos, y el de la conducción automática, cuyo objetivo es mantener la dirección de un vehículo dentro de los márgenes de una carretera y evitar los obstáculos que pudieran aparecer. Dentro de esta última línea, cabe destacar los trabajos de Pomerleau [Pomerleau, 1990; 1993] sobre conducción automática utilizando redes de neuronas.

1.3.3 Enfoques híbridos

En el enfoque reactivo radical se prohibe la construcción de modelos explícitos del entorno [Brooks, 1991], y se rechaza cualquier tipo de planificación global del camino a recorrer. Se acuña el término de *planificación reactiva*, implícita en la codificación de los comportamientos y en el sistema de arbitraje. Sin cuestionar el inmenso valor de las aproximaciones reactivas, sobre todo en lo referente a la construcción de sistemas fiables, robustos y eficientes, hay evidencias de que existen mecanismos biológicos que implementan algún tipo de planificación global, y de que se apoyan en el reconocimiento de ciertos lugares para posteriormente ejecutar el plan. En [Humphreys *et al*, 1992] se relata el caso de un paciente con un pequeña lesión en el lóbulo occipito-temporal del cerebro, encargada del

almacenamiento de la memoria visual. A resultas de esta lesión el paciente era incapaz de reconocer muchos objetos comunes, caras familiares, edificios conocidos, y se encontraba *perdido y desorientado cuando intentaba encontrar un camino en el entorno.*

Los psicólogos se han referido en numerosas ocasiones al problema entorno *versus* cognición como elementos directores de las acciones de un sistema. En [Neisser, 1976, pag. 53] se expone esta dicotomía: *"los conductistas radicales esperan poder explicar la actividad en función únicamente de la estructura del entorno; todas las hipotéticas construcciones explicativas les parecen peligrosamente mentalistas. Las versiones extremas de la Teoría del Procesamiento de la Información o de las teorías constructivistas, por otro lado, no tienen prácticamente en cuenta la información ofrecida por el entorno. Dejan al sistema que percibe perdido en su propio sistema de procesamiento, justo como de las viejas teorías cognitivas se decía que «dejaban a la rata perdida en el pensamiento del laberinto»"*. Podemos identificar el enfoque reactivo como básicamente conductista, mientras que los enfoques clásicos —en los que predomina el desarrollo de modelos matemáticos— presentan similitudes con los enfoques constructivistas, los de la Teoría del Procesamiento de la Información y, en general, con los modelos de la Psicología Cognitiva.

Agre y Chapman [Agre & Chapman, 1990] critican el concepto de planificación reactiva argumentando que es un concepto contradictorio en sí mismo, y defienden que, si bien las ideas tradicionales de planificación necesitan una revisión profunda, el desarrollo de planes es una tarea imprescindible en entornos complejos. Frente al tradicional *plan expresado como programa (plan-as-program)*, analizan nuevas posibilidades, en especial los *planes expresados mediante sugerencias (plan-as-communication[2])*, en los cuales las tareas a realizar no constituyen un conjunto de instrucciones a ejecutar mecánicamente, sino una fuente de apoyo a la toma de decisiones que un agente autónomo deberá desarrollar. En ellos se contempla al sistema como *participando en el mundo, no controlándolo*.

[2] Consideramos que *plan expresado mediante sugerencias* es una traducción más afín a la intención de los autores que la traducción literal "planes como comunicación" que en todo caso pretende subrayar que el planificador comunica al agente (es decir sugiere) la acción a desarrollar.

Según nuestra opinión, las grandes aportaciones de los enfoques reactivos, aunque deben tenerse muy presentes, no deben hacernos olvidar las habilidades demostradas por los métodos clásicos de planificación. Por contra, el enfoque reactivo aporta flexibilidad y dinamismo a las teorías clásicas. Un planteamiento híbrido permite integrar virtudes de ambos enfoques; por este motivo, muchos trabajos actuales [Mataric 1992; Pierce & Kuipers, 1994; Kurz, 1993; 1996; Walker *et al*, 1993; Franchi *et al*, 1994] adoptan posiciones de este tipo. En nuestro trabajo utilizaremos un enfoque híbrido, definiendo inicialmente un subsistema reactivo autónomo, pero sin prescindir posteriormente de la construcción, mantenimiento y utilización de un modelo cognitivo del entorno. El planificador funcionará como un agente más del sistema, proporcionando pautas de comportamiento, en forma de sugerencias o indicaciones, al subsistema reactivo.

1.4 Un enfoque basado en la psicología

Para arrojar luz sobre la ingente cantidad de tareas que debe ser capaz de afrontar un sistema de robótica móvil podemos acudir a las fuentes de la psicología cognitiva. Para esta disciplina, la mente opera en base a una serie de procesos, que son los procesos de *percepción*, *atención*, *aprendizaje*, *memoria*, *pensamiento* y *lenguaje*. Estos procesos, en cooperación, realizan las tareas cognitivas de la mente.

Como ya se ha argumentado, el que la robótica móvil sea un campo de creciente interés para los investigadores en IA se debe a que incorpora prácticamente todos los procesos mencionados. Un sistema de robótica móvil debe dar cuenta de todos ellos si pretendemos que sea autónomo, robusto, fiable y capaz de realizar tareas bajo requerimiento humano. En este apartado analizaremos someramente estos procesos cognitivos desde el punto de vista del desarrollo de sistemas de robótica móvil.

Esta tesis se inspira en cierta medida en la psicología humana. El estudio de la psicología por parte de un investigador en IA aporta ideas y arroja nueva luz sobre algunos conceptos típicos de esta última, que la psicología sitúa en un contexto más amplio aunque al mismo tiempo menos procedimental. En el desarrollo de la memoria de la tesis utilizaremos algunos conceptos tomados de la psicología cognitiva; por este motivo nos ha parecido interesante comparar los procesamientos y necesidades de representación de un robot móvil con algunos

conceptos planteados por la psicología. La amplitud de las relaciones entre robótica móvil y psicología justifica una vez más que la robótica móvil se haya convertido en un campo de pruebas para la IA.

1.4.1 Percepción

El primer problema que hombres y robots tienen que superar es el problema de la percepción. La percepción se entiende como la sensación interna que un evento externo produce en un sistema. Se trata de una información elaborada, y algunos investigadores sostienen [Watt, 1992; Gibson, 1977] que esta elaboración está orientada a la tarea concreta para la que va a ser utilizada. Es decir, un mismo objeto provoca diferentes percepciones si lo que queremos es evitarlo, usarlo o comerlo, por poner algunos ejemplos.

Según este punto de vista, algunos planteamientos iniciales de la robótica móvil eran incorrectos, desde el momento en que los ingenieros pretendían identificar los objetos de una escena como paso previo al cálculo de una ruta libre de obstáculos. En el contexto de la percepción orientada a la tarea da lo mismo que el objeto situado frente al robot sea una mesa, un humano o un armario; lo que interesa es eludirlo. Es mucho más simple transformar una imagen o lectura de sensor en una consigna de "adónde no se debe ir" que identificar a qué tipo de objeto corresponden esas medidas. Es más, en muchos casos, determinar el tipo de objeto no nos ayudará a eludirlo. Si nuestro objetivo es no colisionar, debemos programar algoritmos que transformen nuestras medidas sensoriales en información útil para este propósito, y olvidarnos del resto. Con un objetivo distinto en mente habría que diseñar diferentes algoritmos de percepción.

La percepción orientada a la tarea, junto a los planteamientos reactivos, ha sido sometida a fuertes críticas por parte de los cognitivistas radicales [Pylyshyn, 1988], pero es ineludible que su uso ha proporcionado grandes avances en la construcción de sistemas autónomos, y por tanto debe tenerse muy presente como metodología de trabajo en este campo.

Hay que dedicar también un comentario a la *sensibilidad visceral* del robot. El robot no sólo siente eventos externos. En muchos casos los robots van equipados con sensores que informan del nivel de carga de las baterías o del mal funcionamiento de algún dispositivo.

Esta sensibilidad interna o visceral ha sido abordada por Arkin [Arkin, 1993], quien argumenta que es fundamental para desarrollar sistemas verdaderamente autónomos que estos puedan dar cuenta de sus propias necesidades de mantenimiento, en lo que denomina *control homeostático* del robot.

Finalmente, otro elemento que también puede integrarse dentro del contexto reactivo es la motivación. Las aspiraciones o metas que el robot desea alcanzar generan impulsos que pueden convertirse en comportamientos o tendencias de movimiento. Estas metas pueden ser autoimpuestas por el control homeostático (p. ej. cargar las baterías), oportunistas (p. ej. recoger un bote de refresco encontrado en el suelo) u obedecer a órdenes de usuario (p. ej. recoger un paquete en conserjería).

1.4.2 Sensomotricidad

Ya hemos comentado anteriormente la vinculación entre los planteamientos reactivos y los esquemas sensoriomotrices de Piaget; en los seres vivos existen comportamientos que se manifiestan en forma de *arcos reflejos*, es decir, no requieren de un procesamiento elaborado de la información sensorial. Estos comportamientos conectan directamente la información sensorial con el sistema motor. Es lógico pensar que, si pretendemos que un robot se desenvuelva con soltura en un medio no estructurado, debemos equiparlo con comportamientos reactivos, o arcos reflejos, que eviten colisiones imprevistas provocadas, por ejemplo, por obstáculos móviles. Los arcos reflejos incrementan la robustez y fiabilidad de los robots y, además, descargan de trabajo al sistema de proceso [Mira-Mira, 1996].

1.4.3 Memoria, aprendizaje y resolución de problemas

La memoria ocupa un papel protagonista en los estudios de los psicólogos cognitivos; por supuesto memoria en este contexto no se refiere sólo al acceso a los datos, sino a una recuperación inteligente de estos, lo que se conoce como *memoria asociativa*.

Los psicólogos distinguen entre tres tipos de memoria [Lindsay & Norman, 1983]: el *almacén de información sensorial, o imagen sensorial*, la *memoria a corto plazo*, y la *memoria a largo plazo*.

En esta tesis se propone una arquitectura fuertemente centrada en la memoria. Como veremos en el capítulo 6, podemos establecer estrechas relaciones entre los tres tipos de memoria identificadas por los psicólogos y los tipos de memoria que aparecerán en los sistemas desarrollados. Los procesos se conciben como algoritmos concurrentes que transforman unos tipos de memoria en otros, y cuyo resultado físico es un comando de movimiento que se enviará al robot.

1.4.4 Planificación de tareas

Hemos defendido en esta introducción que un robot móvil no debe basarse exclusivamente en una colección de comportamientos reactivos. Los comportamientos reactivos, o arcos reflejos, son necesarios porque ayudan al sistema a resolver imprevistos y facilitan la integración entre los sensores y los motores (y en esta tarea los sistemas tradicionales han demostrado ser bastante deficitarios), pero no son suficientes, porque no dan respuesta adecuada al problema del conocimiento incremental del medio y al del uso de este conocimiento para obtener mejores soluciones a los problemas. Para superar estos inconvenientes es necesario regresar a un conocido y viejo concepto: la *planificación de tareas*. Asumiremos, sin embargo, que este concepto necesita una profunda revisión: cuando el ejecutivo del plan es un sistema reactivo, que dispone de mayores habilidades que los ejecutivos de planes tradicionales, la noción de plan ya no es la misma. Bajo estos supuestos, e inspirándose en las propuestas de Agre y Chapman [Agre & Chapman, 1990], Payton [Payton, 1991] concibe la idea *de plan internalizado*. Un plan internalizado consiste en una especie de red de sugerencias o “pistas” que el planificador proporciona al robot en ciertos lugares del entorno. El agente reactivo deberá interpretar estas sugerencias teniendo en cuenta las particularidades presentes en el momento de utilizar el plan, y observar los imprevistos que pudieran aparecer.

1.4.5 Lenguaje

El apartado de comunicación con el usuario es un tema poco tratado en los trabajos actuales. No obstante, si queremos que robots y humanos cooperen para desarrollar objetivos comunes, es necesario que dispongan de un sistema elaborado de comunicación. Este lenguaje común debe permitir, al menos, que el humano pueda especificar objetivos al robot y que el robot pueda comunicar al usuario los descubrimientos realizados en el curso de una misión.

1.5 Objetivos y plan de trabajo

Esta tesis pretende enfrentar la construcción de una arquitectura para un robot móvil dentro de un enfoque híbrido. En concreto, nuestros objetivos serán los siguientes:

- Definir y desarrollar una serie de técnicas de tipo reactivo que permitan a un robot móvil, utilizando información sensorial (básicamente sensores de ultrasonidos, infrarrojos y/o táctiles), desenvolverse en un medio desconocido a priori: no colisionar con los objetos del entorno, resolver misiones simples, eludir obstáculos, etc.
- Definir y desarrollar técnicas de modelado capaces de integrar en estructuras cognitivas la información sensorial captada por el robot. Estas técnicas posibilitarán el modelado incremental, refinando el conocimiento a medida que el sistema recibe nueva información del entorno.
- Definir estrategias de planificación capaces de explotar los modelos cognitivos creados.
- Definir y desarrollar una arquitectura para un robot móvil que permita el uso conjunto de la información sensorial y del modelo cognitivo para resolver misiones complejas.
- Integrar los algoritmos desarrollados en una plataforma móvil NOMAD-200, y validar experimentalmente su correcto funcionamiento.

Como metodología de trabajo adoptaremos la propuesta de Arkin [Arkin, 1993], donde recomienda la implementación previa de los algoritmos en un simulador. De este modo las técnicas en consideración pueden ser probadas en condiciones “de laboratorio” antes de ser portadas al robot real. Si los métodos son robustos, los problemas que surjan sobre el robot real (ruido en los sensores, imprecisión en el posicionamiento, etc.) podrán ser resueltos con pequeñas modificaciones al modelo inicial. De lo contrario el modelo se rechazará.

Los diferentes niveles de funcionamiento se abordarán por separado, con la intención de que puedan ser reutilizados en otros trabajos. De este modo, el subsistema reactivo —por ejemplo— será independiente de los sistemas de construcción y explotación del modelo cognitivo, y podrá utilizarse en otras aplicaciones con diferente sistema de modelado o incluso

sin modelado en absoluto, en un enfoque puramente reactivo. Así mismo, las técnicas de modelado serán utilizables con diferentes subsistemas reactivos, ya que simplemente proporcionarán consignas de navegación —a modo de indicaciones o sugerencias— al nivel reactivo, que éste interpretará de acuerdo a su propia naturaleza.

Cada uno de los objetivos enunciados aquí se refleja en uno o varios capítulos de esta memoria; del nivel reactivo se ocupa el capítulo 2, mientras que del desarrollo del modelo cognitivo (eje central de esta tesis) se ocupan los capítulos 3 y 4. En el capítulo 5 se trata el problema de la explotación de la información recogida en los mapas cognitivos, mientras que en el capítulo 6 se expone la arquitectura en que se integran todos los elementos anteriores. El capítulo 7 trata sobre diseño e implementación *software*, y finalmente el capítulo 8 se refiere a las pruebas experimentales realizadas.

Este capítulo de introducción y un capítulo 9 de conclusiones y desarrollo futuro completan la estructura de la memoria.

2. El nivel reactivo: subsistema de navegación

En este nivel de abstracción, nuestro propósito será evitar que el robot colisione con objetos tanto estáticos como móviles y resolver problemas sencillos de planificación, tales como navegar siguiendo un determinado rumbo o aproximarse a ciertas coordenadas espaciales. El robot no poseerá ningún modelo previo del entorno en el que se mueve; utilizará la información suministrada por sus sensores para decidir en cada momento la dirección a tomar. Se pretende en este nivel sentar las bases para un comportamiento global robusto y fiable. Nuestro enfoque para el nivel de navegación está bastante relacionado con los planteamientos de control reactivo de Arkin [Arkin, 1990; Arkin, 1993; Arkin, 1994].

En capítulos posteriores se definirán métodos de construcción de modelos cognitivos que dotarán al sistema de diversas capacidades de aprendizaje del entorno de trabajo. Las operaciones de planificación sobre estos modelos descompondrán el problema en forma de subtareas simples capaces de ser resueltas por el nivel que ahora nos ocupa.

Aún por debajo de este nivel existe otro más elemental, tradicionalmente denominado nivel de pilotaje, integrado por un conjunto de procesos de control que gestionan las órdenes de dirección del navegador y envían comandos de movimiento a los motores, evitando cambios bruscos de dirección y velocidad que podrían dañar al *hardware*. El nivel de pilotaje no es objetivo de este trabajo, si bien ha sido necesario, para poder desarrollar y probar el resto de niveles, implementar un pequeño sistema de control que cumpla estos requerimientos, tal y como se comentará en el capítulo 7, dedicado a diseño e implementación.

En este capítulo utilizaremos con frecuencia el concepto de imagen sensorial. Como se comentó en la introducción, la imagen sensorial es una transformación de las medidas de los sensores externos del robot (ultrasonidos, infrarrojos, visión, etc.) en un conjunto de valores que representan la distancia al objeto más próximo en una serie de direcciones. En algunos casos, como sucede con los sensores de ultrasonidos, la obtención de la imagen sensorial es inmediata. En otros, como la visión estereoscópica, puede ser necesario un complejo proceso de cálculo.

2.1 El método clásico del potencial

En el método del potencial artificial [Khatib, 1986], el robot se concibe como una partícula inmersa dentro de un campo de potencial cuyas variaciones locales reflejan la estructura del entorno. Los obstáculos son modelados por cargas repulsivas y la meta como una carga atractiva. El movimiento del robot se establece de modo iterativo, calculando en cada paso la fuerza generada por el campo y utilizando la dirección de ésta para modificar el comportamiento del robot. Aunque la formulación original de la teoría se ocupaba del problema de la planificación de trayectorias en robots articulados, la idea fundamental, con algunas modificaciones, puede aplicarse al problema de la navegación de robots móviles.

2.2 Método del potencial en entornos desconocidos

En nuestro trabajo [Serradilla & Maravall, 1996] hemos utilizado la imagen sensorial para obtener en tiempo real una estimación de la dirección de movimiento del robot. Otros investigadores [Krogh, 1984, Brooks, 1986; Arkin, 1989] han utilizado con anterioridad enfoques similares a éste, aunque sorprendentemente se ha dedicado mucha más atención al problema de la planificación con campos de potencial en entornos conocidos, es decir, modelando los objetos del entorno y calculando *off-line* la trayectoria a seguir.

En nuestro trabajo utilizamos la información sensorial, junto con la dirección deseada de movimiento, para determinar la magnitud de la fuerza repulsiva que en cada momento deberá actuar sobre el robot, traducida a comandos de velocidad. Este enfoque presenta varias ventajas:

- No es necesario un modelo previo del entorno.

- El sistema es altamente robusto: las modificaciones del entorno no afectan al rendimiento del sistema.
- Se permite la existencia de objetos móviles.
- El planteamiento es adecuado para la planificación en tiempo real.
- Las trayectorias obtenidas son localmente óptimas.

2.2.1 Rumbos y metas

Dispondremos de tres mecanismos para comunicar al sistema nuestros deseos de movilidad. En general, los módulos de más alto nivel comunicarán deseos de movilidad al nivel de navegación a través de alguno de estos tres mecanismos. Estos mecanismos son los siguientes:

1. *Modo meta*: existen unas coordenadas objetivo que queremos alcanzar. Cada cierto intervalo de tiempo el sistema revisa el rumbo para hacer que apunte hacia la dirección objetivo. Para poder realizar este cálculo es preciso conocer, al menos de modo aproximado, la posición del robot.
2. *Modo rumbo básico*: existe una tendencia o dirección preferida de movimiento. Digamos que establece algo así como el punto cardinal hacia el que queremos desplazarnos.
3. *Modo rumbo con cambio suave*: en esencia es similar al anterior, salvo que introduce un cierto retardo en el cambio de rumbo de manera que el vector que expresa el rumbo rota suavemente hasta alcanzar el rumbo deseado. Puesto que en esencia no es diferente al anterior, nos referiremos a ambos genéricamente como modo rumbo.

En el modo rumbo, la dirección de navegación deseada se representa por un vector $\vec{G}$ unitario y constante proporcionado por el usuario o por un módulo de mayor abstracción. En el modo meta, $\vec{G}$ no es constante, sino que se recalcula periódicamente según la ecuación 2.1.

$$\vec{G} = \vec{p}_g - \vec{p}_r \qquad (2.1)$$

donde $\vec{p}_g$ es la posición de la meta y $\vec{p}_r$ es la posición actual del robot.

Es importante observar que, mientras que el modo rumbo es sensible a errores en la orientación del robot, el modo meta es sensible tanto a errores de orientación como a errores de posición. Esta sensibilidad a errores de orientación y posicionamiento aparece en todo sistema que tenga como objetivo alcanzar un lugar del mundo especificando sus coordenadas geométricas. Los problemas de posicionamiento y orientación son problemas a los que, de un modo u otro, tienen que enfrentarse todos los investigadores que trabajan en este campo.

Desde las perspectivas más clásicas, el problema se soluciona añadiendo al entorno un sistema de balizado que permita en todo momento determinar con exactitud las coordenadas del robot. En las aproximaciones reactivas el problema se evita estableciendo objetivos no geométricos que sean detectables sensorialmente (p. ej. buscar la luz) o no estableciendo objetivos en absoluto. En las aproximaciones híbridas se permite definir objetivos geométricos aproximados que se complementan con información sensorial local, lo que da lugar a mecanismos robustos sin perder la potente funcionalidad de referirse a los objetivos a través de su posición geométrica.

En este trabajo hemos optado por la tercera línea, apoyándonos en varios elementos que serán desarrollados mas adelante. Estos elementos son:

- Utilizar el modo rumbo siempre que sea posible, puesto que es menos sensible a la acumulación de errores.
- Almacenar en ciertos lugares del entorno, detectables sensorialmente, la posición aproximada de ese lugar junto con una imagen sensorial promedio, lo que permitirá validar que el lugar encontrado es el esperado.
- Descomponer las misiones en pequeñas tareas de carácter local, que serán realizadas preferentemente en modo rumbo.

2.2.2 Repulsión debida a los obstáculos

Hemos visto el primer requisito que debe cumplir el subsistema de bajo nivel: intentar aproximarse al objetivo. El segundo requisito fundamental es el de evitar las colisiones con los objetos del entorno. La teoría de campos de potencial artificial proporciona un método elegante para integrar en una única ecuación ambas necesidades. Si el objetivo funciona como

una carga de distinto signo a la que representa al robot, generando una fuerza atractiva hacia él, los obstáculos encontrados van a funcionar como cargas del mismo signo que la del robot, generando fuerzas repulsivas que tenderán a alejar al robot de los objetos con los que pudiera colisionar.

Para estimar la fuerza repulsiva debida a los obstáculos tan sólo disponemos de la información suministrada por los sensores; debido a la hipótesis de mundo desconocido, no conocemos la forma ni la disposición de los objetos ni de las paredes. Para estimar la fuerza a aplicar a partir de la información sensorial, consideraremos que cada sensor debe generar una fuerza opuesta a la orientación del sensor y de magnitud inversamente proporcional a la lectura del mismo. Es decir, mientras más cerca se encuentre el robot del objeto tanto mayor será la fuerza aplicada en sentido contrario. Este cálculo se resume en la ecuación 2.2.

$$\vec{S} = \sum_i -\frac{k_s}{r_i^{m_s}} \vec{o}_i \qquad (2.2)$$

donde k_s y m_s son constantes que modelizan la magnitud de la repulsión en función de la distancia, r_i es la medida de rango proporcionada por el sensor i y $\vec{o}_i$ es un vector unitario en la dirección del sensor i. En las pruebas experimentales hemos utilizado valores de $k_s = 8$ y $m_s = 1{,}2$. Posteriormente trataremos en profundidad el problema del ajuste de las constantes k_s y m_s.

Como se muestra en la ecuación 2.3, la fuerza resultante será la suma de las dos componentes (figura 2.1).

$$\vec{F} = \vec{G} + \vec{S} \qquad (2.3)$$

Integrando la fuerza $\vec{F}$ para todos los puntos del entorno (esto sólo es posible en un mundo simulado, ya que tenemos que situar al robot en cada punto del mundo para calcular $\vec{F}$) obtenemos el campo de potencial asociado a dicho entorno. El

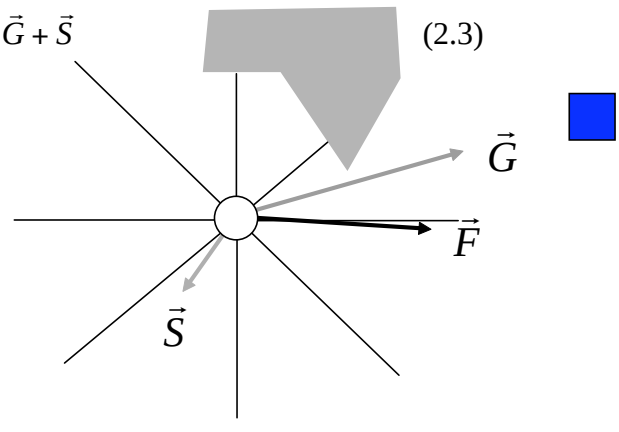


Figura 2.1. Fuerzas básicas para calcular la trayectoria de navegación del robot

campo de potencial ilustra bastante bien el proceso de cálculo de trayectorias porque establece un símil entre dicho proceso y la trayectoria que seguiría una esfera que pudiera rodar libremente sobre la superficie del campo siguiendo siempre la dirección de máxima pendiente (ignorando los momentos de inercia). Podemos establecer la trayectoria que seguiría el robot desde cualquier punto de origen calculando iterativamente la dirección de máxima pendiente y desplazando al robot en dicha dirección; es decir, el robot seguiría una línea de campo que pasara por su punto de origen.

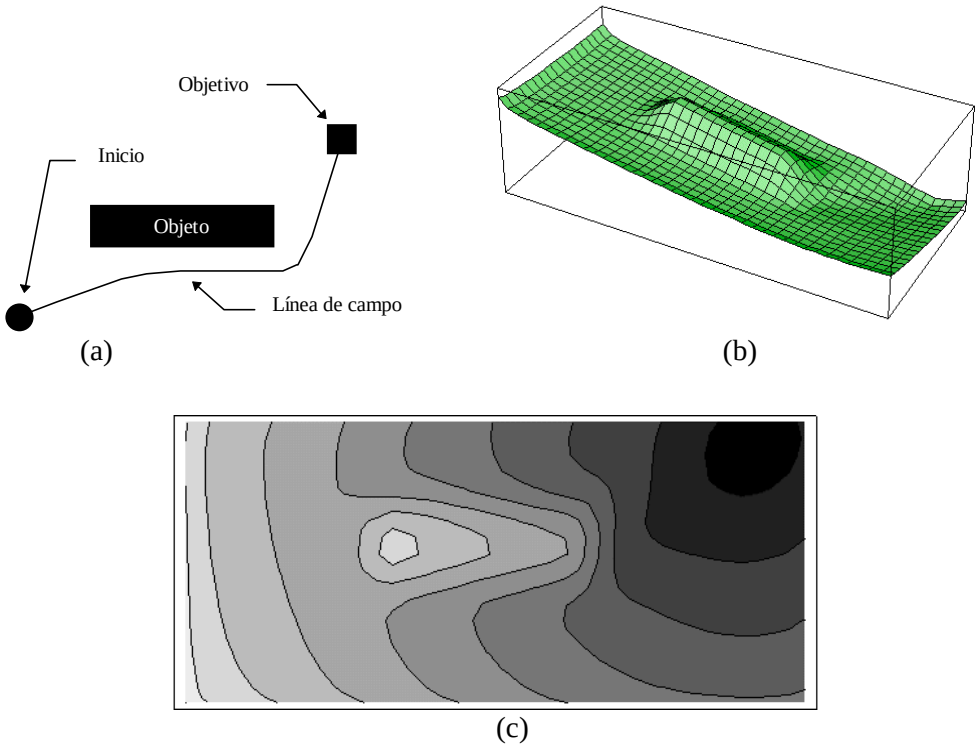


Figura 2.2. Campo generado por el método del potencial en un problema simple. (*a*) *entorno del robot;* (*b*) *superficie del campo de potencial;* (*c*) *curvas de nivel del campo de potencial*

En la figura 2.2 hemos representado (a) el entorno del robot, (b) el campo de potencial asociado y (c) un gráfico de curvas de nivel del campo de potencial.

2.3 Problemas de los métodos de potencial

Un serio problema del enfoque tradicional, ya detectado por sus creadores [Khatib, 1986], es su elevada sensibilidad a la existencia de mínimos locales. Estos mínimos, además, no sólo son relativamente frecuentes sino que dependen de cada misión particular, por lo que no es posible su prevención. Históricamente se han propuesto varios métodos para escapar de ellos

una vez que son detectados, descritos en [Latombe, 1993]. En general, estos métodos implican un mecanismo de exploración costoso, tanto en tiempo de proceso como en memoria necesaria para almacenar un modelo adicional de la situación de los mínimos locales. Un problema añadido consiste en el mantenimiento de este nuevo modelo en entornos cambiantes.

La existencia de un mínimo local depende de la estructura del entorno en conjunción con la posición de la meta, como puede observarse en las figuras 2.3 a 2.7. Sin embargo es completamente independiente de la posición inicial del robot, la cual sólo determinará si el robot cae o no en el mínimo.

2.4 Método de las cargas ficticias

Con objeto de paliar los problemas surgidos con la aparición de mínimos locales, proponemos en esta tesis un nuevo método, que hemos denominado *método de las cargas ficticias*. Este método consiste en dotar al sistema con la capacidad de añadir cargas repulsivas ficticias que obliguen al robot a alejarse de los mínimos locales una vez que son detectados.

La situación de mínimo local puede detectarse teniendo en cuenta la suma total de las fuerzas que inciden sobre el robot. Cuando esta fuerza se hace muy pequeña el movimiento inducido debería ser prácticamente nulo: estamos pues en un mínimo local cuando la fuerza ejercida sobre el robot es menor que cierto valor umbral ξ.

Figura 2.3. Una misión en la que aparece un mínimo local en el campo de potencial

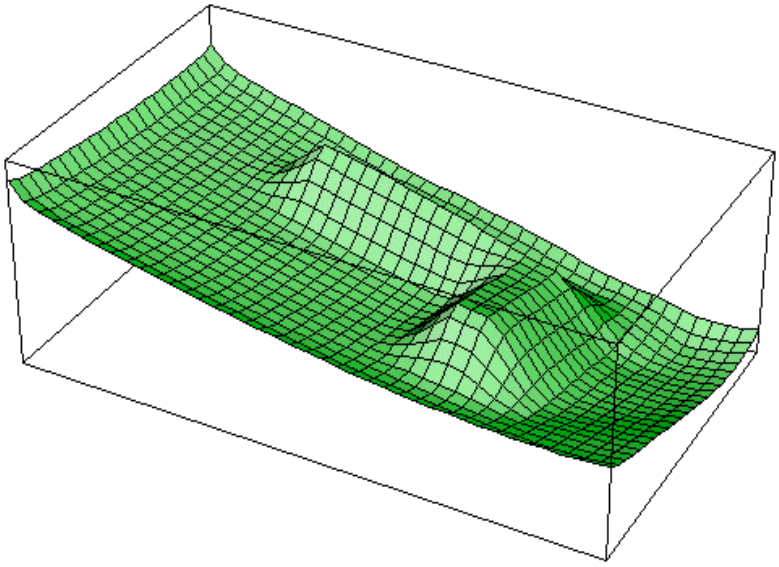


Figura 2.4. Campo generado en el problema de la figura 2.3

Figura 2.5. Otra misión en el mismo entorno en la que no aparece mínimo local en el campo de potencial

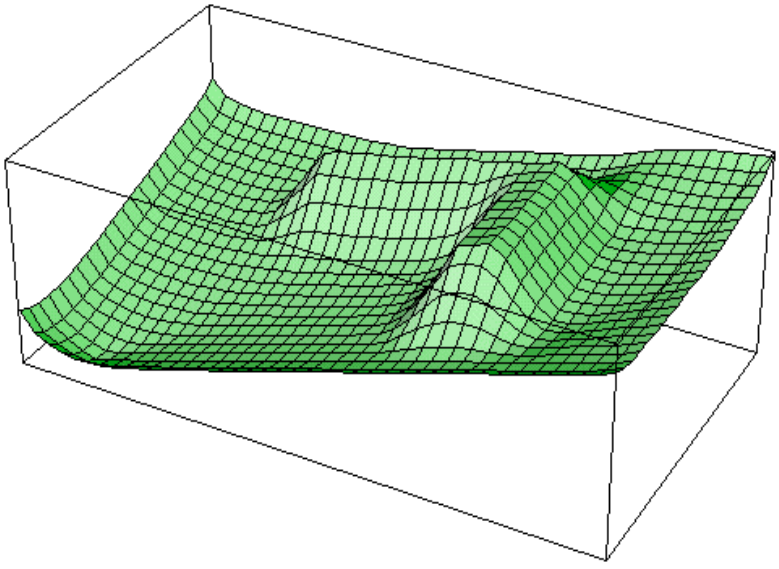


Figura 2.6. Campo generado en el problema de la figura 2.5

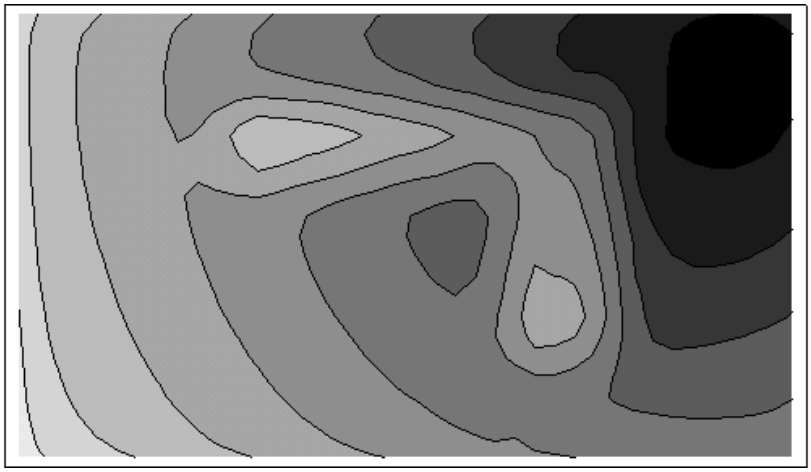


Figura 2.7. Gráfico de curvas de nivel en el que se aprecia la existencia de un mínimo local en la función de campo de la figura 2.4

Ante una situación de este tipo colocaremos una carga repulsiva unitaria que incite al robot a escapar del mínimo, en el punto determinado por la ecuación 2.3.

$$\vec{p}_c = \psi(\vec{p}_r - \vec{S} + \vec{l}) \qquad (2.3)$$

donde $\vec{p}_r$ es la posición actual del robot, $\vec{S}$ es la fuerza de repulsión asociada a los objetos y $\vec{l}$ es un vector unitario perpendicular al vector que apunta hacia la meta (vector $\vec{G}$). Ψ es un número real que determina la distancia a la que se colocará la carga; en nuestro trabajo hemos utilizado una constante Ψ ajustada experimentalmente a un valor de 2 radios de robot[3]. En la figura 2.8 se muestra un ejemplo de colocación de una carga ficticia con $\Psi = 2$.

Hay que llamar la atención sobre el hecho de que en el plano existen dos vectores perpendiculares a $\vec{G}$ según tomemos la perpendicular por la derecha o por la izquierda, lo que genera, a su vez, dos posibles comportamientos para el robot. Para la elección del vector perpendicular tenemos diversas opciones:

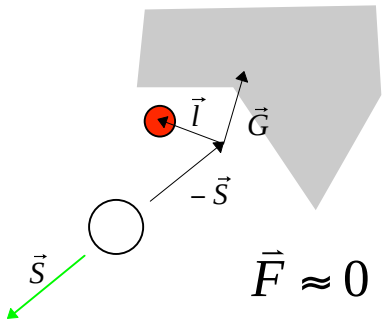


Figura 2.8. Colocación de una carga por un robot diestro

1. Siempre a la derecha (el robot es zurdo, porque

[3] El robot NOMAD-200 posee un radio de 9 pulgadas, de modo que, para las pruebas experimentales, Ψ=18 pulgadas.

tendería en los mínimos locales a moverse hacia la izquierda).

2. Siempre a la izquierda (el robot es diestro).
3. Aleatoriamente a derechas o a izquierdas. No da buen resultado, porque tiende a bloquear al robot cuando se colocan varias cargas consecutivas.
4. Aleatoriamente a derechas o a izquierdas, en ráfagas. Es decir, cuando se decide una orientación se mantendrá durante cierto número de ensayos.
5. A izquierdas o a derechas dependiendo de los últimos movimientos del robot, siguiendo la trayectoria *más natural*. Por más natural entendemos la que menos difiera en ángulo la dirección actual del robot.

Después de una serie de pruebas este último método demostró ser el más robusto, por lo que se ha adoptado para la implementación final.

Cuando se han añadido una o varias cargas, la fuerza total debida a cargas ficticias obedece a la expresión descrita en la ecuación (2.4).

$$\vec{C} = \sum_i - \frac{k_c}{\left|\vec{C}_i\right|^{m_c}} \vec{c}_i \qquad (2.4)$$

donde $\left|\vec{C}_i\right|$ es la distancia del robot a la carga *i* y $\vec{c}_i$ es un vector unitario en la dirección de la carga *i*. De nuevo dos parámetros, k_c y m_c, establecen cómo decae la repulsión con la distancia. Lo ideal sería que k_c fuera proporcional a la magnitud del mínimo local, aunque dicha magnitud a priori se desconoce. Actualmente utilizamos valores —ajustados experimentalmente— de k_c=8 y m_c=1.

$$\vec{F} = \vec{G} + \vec{S} + \vec{C} \qquad (2.5)$$

Tras la incorporación de cargas ficticias a nuestro modelo, las fuerzas que intervienen son las mostradas en la ecuación 2.5 (ver figura 2.9).

La adición de una o varias cargas ficticias equivale a provocar artificialmente una elevación en la superficie del campo de potencial. Con ello conseguimos de modo natural que el robot evite el mínimo, sin tener que recurrir a cambios de comportamiento o a la ejecución de algoritmos alternativos, como sucede en las propuestas tradicionales. En las figuras 2.11 y 2.12 puede verse el efecto de la colocación automática, una vez alcanzado el mínimo local justo en el vértice cóncavo del objeto, de varias cargas ficticias para escapar del mínimo mostrado en la figura 2.7. Las cargas colocadas y la trayectoria descrita por el robot se muestran en la figura 2.10.

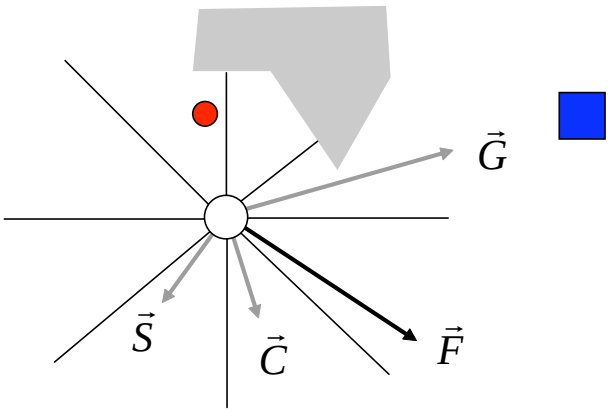


Figura 2.9. Fuerzas básicas para calcular la trayectoria de navegación del robot incorporando cargas ficticias. El círculo oscuro representa una carga ficticia.

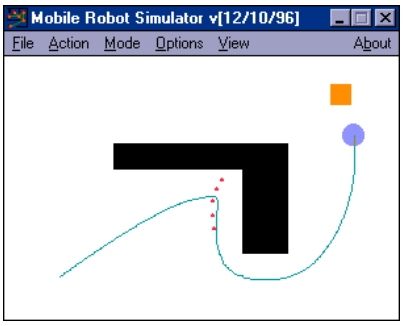


Figura 2.10. Cargas añadidas para escapar del mínimo local y trayectoria descrita por el robot.

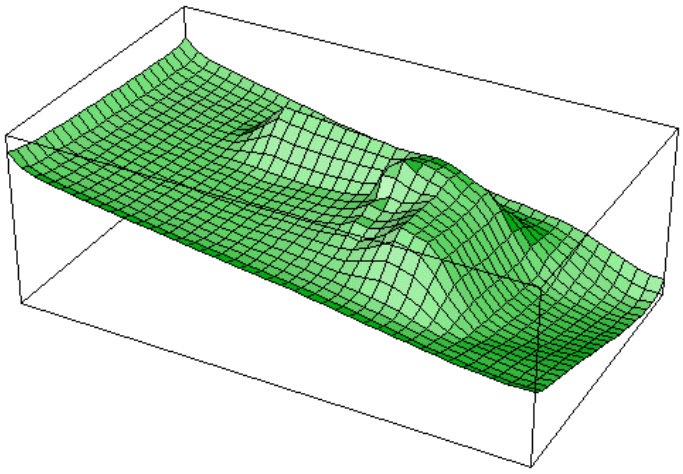


Figura 2.11. Superficie modificada mediante la adición de varias cargas ficticias

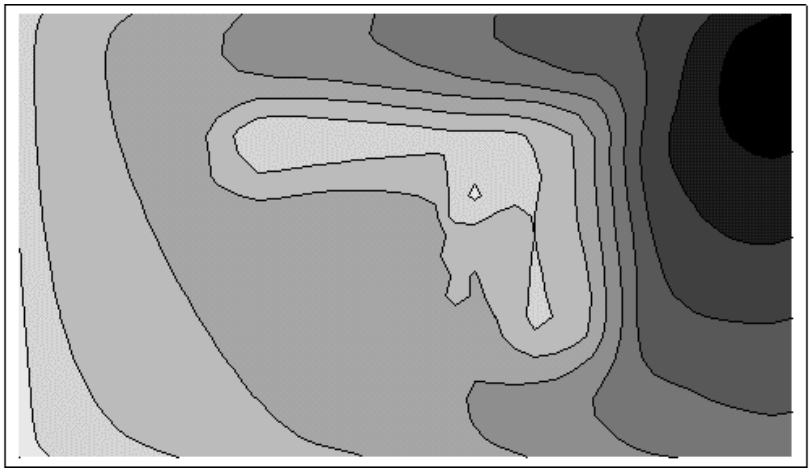


Figura 2.12. Gráfico de curvas de nivel donde se aprecia la desaparición del mínimo local en la superficie del campo de potencial gracias al uso de cargas ficticias

Latombe alude a una hipotética función de potencial —la función de navegación— que, de encontrarse, permitiría la navegación hasta el objetivo aplicando simplemente un criterio de descenso del gradiente [Latombe, 1993]. Aunque esta función no existe en general, el método de las cargas ficticias obtiene una buena aproximación de la misma, corrigiendo el campo de potencial artificial tradicional con la adición de nuevos términos representados mediante las cargas ficticias.

2.5 Ajuste de los valores de repulsión

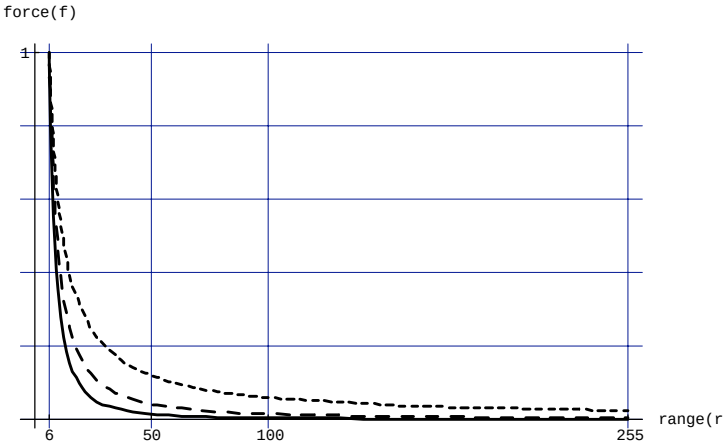


Figura 2.13. Curvas de ajuste de los valores de repulsión.

Hemos visto que la función que expresa cómo decae la fuerza repulsiva con la distancia depende de dos parámetros *k* y *m*. En la figura 2.13 se muestran tres posibles curvas, correspondientes a pares de valores *k*=6 y *m*=1 (en líneas discontinuas cortas), *k*=14,7 y *m*=1,5 (en líneas discontinuas largas), *k*=35 y *m*=2 (en línea continua). A medida que el exponente *m* crece, la fuerza repulsiva decrece más rápidamente con la distancia, es decir, los objetos más alejados influyen

cada vez menos en el comportamiento del robot. Para un valor de *m* prefijado, el parámetro *k* determina la magnitud de la fuerza a una distancia dada. A partir de la ecuación que establece la intensidad de la repulsión de cada sensor individual,

$$f = \frac{k}{r^m} \tag{2.6}$$

podemos obtener dos expresiones que nos permiten calcular *k* y *m* dados dos pares de valores para *f* y *r*. Siendo f_1 la fuerza deseada para una distancia r_1 y f_2 la fuerza deseada para una distancia r_2, según la expresión 2.6, tenemos

$$f_1 = \frac{k}{r_1^m} \tag{2.7}$$

$$f_2 = \frac{k}{r_2^m} \tag{2.8}$$

tomando logaritmos en las expresiones 2.7 y 2.8

$$\ln(f_1) = \ln\left(\frac{k}{r_1^m}\right) \tag{2.9}$$

$$\ln(f_2) = \ln\left(\frac{k}{r_2^m}\right) \tag{2.10}$$

y operando

$$\ln(f_1) = \ln(k) - m\ln(r_1) \tag{2.11}$$

$$\ln(f_2) = \ln(k) - m\ln(r_2) \tag{2.12}$$

Finalmente, restando las expresiones 2.11 y 2.12, llegamos a

$$\ln(f_1) - \ln(f_2) = m \cdot (\ln(r_2) - \ln(r_1)) \tag{2.13}$$

y volviendo a operar

$$m = \frac{\ln(f_1 / f_2)}{\ln(r_2 / r_1)} \qquad (2.14)$$

Una vez calculado *m*, podemos obtener *k* de la ecuación 2.7

$$k = f_1 \cdot r_1^m \qquad (2.15)$$

Estas dos ecuaciones son útiles puesto que nos permiten obtener la curva de rechazo dependiendo de las características particulares de los sensores, fundamentalmente el alcance mínimo y el alcance máximo del sensor, y de las fuerzas deseadas en estos puntos. Interesa en particular que las fuerzas generadas en los alcances máximos sean pequeñas, y que las fuerzas generadas en los alcances mínimos sean de magnitud suficiente para compensar la fuerza de atracción hacia la meta (que consideramos unitaria), ya que de lo contrario el robot podría colisionar. Las tres curvas de la figura 2.11 han sido ajustadas para que la fuerza en el alcance mínimo del sensor (6 pulgadas en nuestros sensores) sea igual a 1.

2.6 Reflexiones sobre alcance, número y duración temporal de las cargas

En torno al uso de cargas ficticias surgen varios problemas interesantes. En primer lugar, ¿qué alcance debe tener una carga ficticia? O reformulada la pregunta de un modo más técnico: ¿cuáles deben ser los valores de los parámetros *k* y *m* para el cálculo de la magnitud de la fuerza repulsiva ejercida por la carga? Puesto que el objeto de la carga es evitar un mínimo local, parece obvio que la influencia repulsiva ejercida por la carga debería decaer rápidamente con la distancia. De no ser así, su efecto podría sentirse lejos del mínimo y afectar a la trayectoria ulterior del sistema. Sin embargo, debe decaer más lentamente que la repulsión generada por los obstáculos, porque de lo contrario el robot apenas se alejaría de la carga cuando estuviera entre varios obstáculos. Es interesante plantear la posibilidad de utilizar diversos alcances en función de la magnitud del mínimo encontrado, lo cual nos lleva a una nueva cuestión: ¿cómo podemos estimar la magnitud de un mínimo a partir de las medidas de los sensores?

En segundo lugar, cabe preguntarse si debe existir alguna limitación al número de cargas ficticias activas en un instante dado. Si este número no se limita, aparecen dos problemas, (a) la memoria necesaria en una misión no está acotada y (b) si el número de cargas crece

demasiado la influencia de éstas domina sobre las fuerzas ejercidas por la meta y por los obstáculos, hecho no deseable en absoluto. En nuestro trabajo hemos optado por limitar el número de cargas utilizables. Esto permite evitar los dos problemas descritos aunque tiene un inconveniente: existe un límite para el tamaño máximo de mínimo local que el sistema puede eludir. En la figura 2.9 tenemos un ejemplo de mínimo local de gran envergadura donde el sistema de navegación queda atrapado. No obstante, pensamos que pretender que el navegador sea capaz de solventar por sí solo un problema excesivamente complejo excede su nivel de competencia. Para ello se definirán los mecanismos apropiados en los capítulos siguientes.

Una idea interesante para superar el problema de la limitación al número de cargas es trabajar con cargas cuya intensidad decaiga no sólo con la distancia, sino también con el tiempo transcurrido desde su fijación. En cualquier caso, las cargas no deben recordarse permanentemente, ya que los mínimos locales son eventos dependientes de la posición de la meta, y por ello de cada misión particular. Además, puesto que son sensibles a la acumulación de errores en el posicionamiento del robot, su ámbito debe ser local. En nuestro sistema, cada vez que una misión local concluye con éxito, todas las cargas ficticias utilizadas se eliminan.

Desde una perspectiva práctica, el problema se resuelve bastante bien utilizando cargas fijas de magnitud unitaria, parámetros k_2=8 y m_2=1, y un número limitado y pequeño de cargas, en nuestro caso 5 como máximo. Esta solución obvia el problema de la estimación del tamaño del mínimo gracias a la colocación sucesiva de varias cargas si el mínimo es grande.

2.7 Ejemplos de navegación utilizando el método del potencial en tiempo real con cargas ficticias

En este apartado incluimos algunos ejemplos de simulación del funcionamiento del navegador descrito (figura 2.14). En ellos se muestra la habilidad del método para (a) encontrar rutas libres de colisiones, (b) evitar obstáculos móviles, (c) y (d) eludir mínimos de magnitud media en la función de potencial.

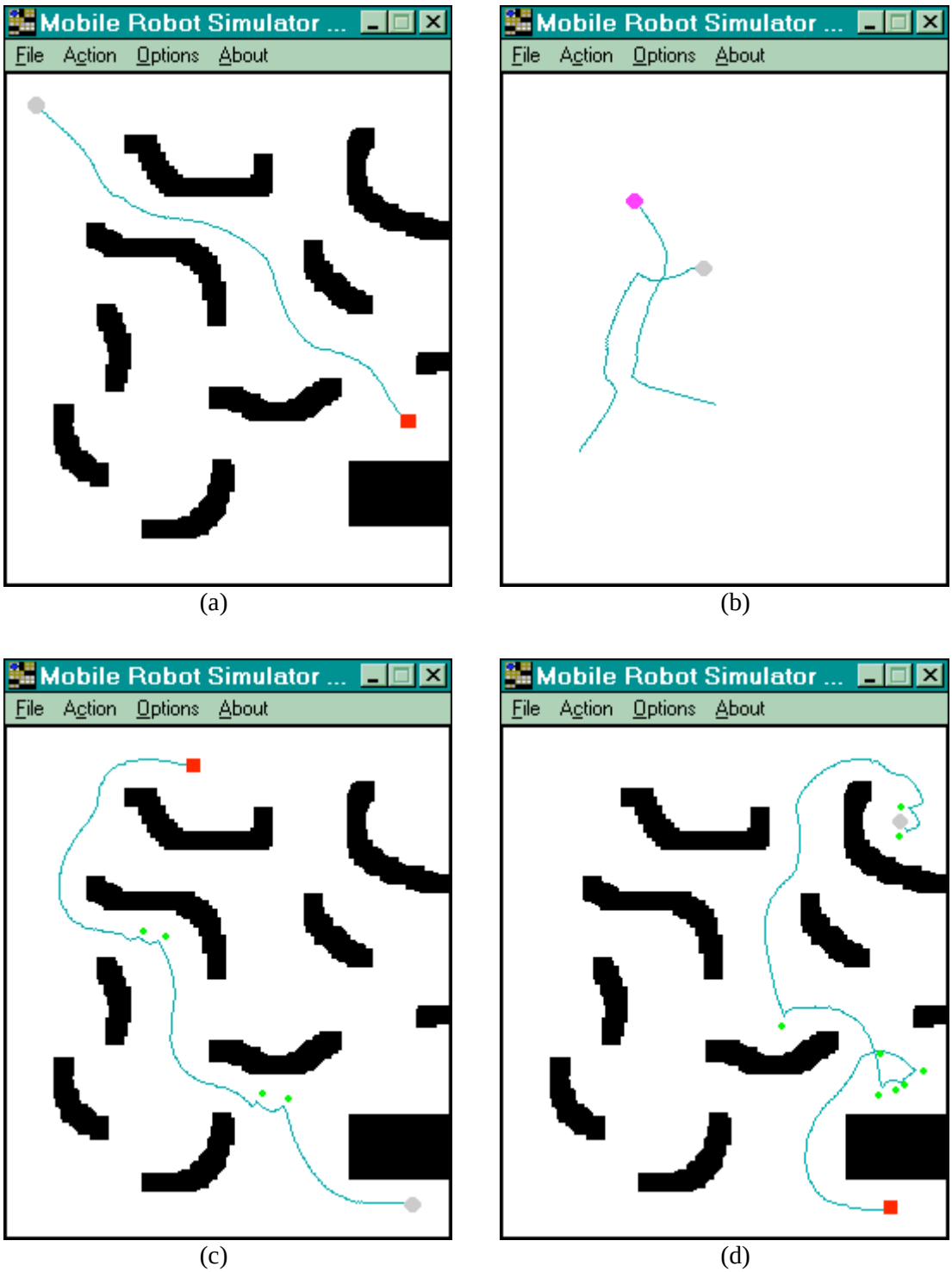


Figura 2.14. Ejemplos de navegación: (*a*) *ruta sin mínimos locales;* (*b*) *evitación de un obstáculo móvil,* (*c*) *y* (*d*) *ruta con mínimos locales y cargas ficticias*

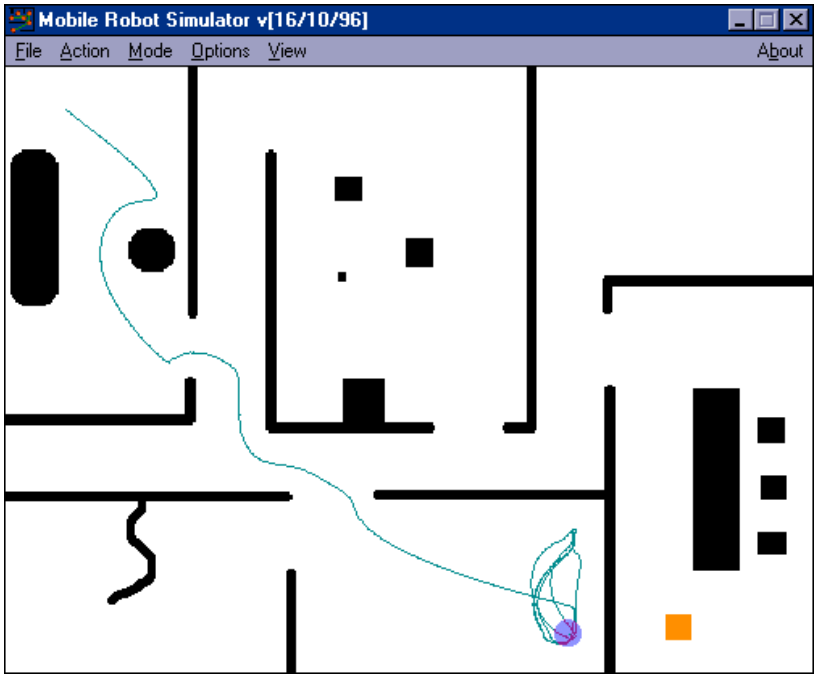


Figura 2.15. Robot atrapado en un mínimo local de gran magnitud

Insistimos en que el objetivo de este nivel no es el de resolver por sí mismo problemas planteados en entornos complejos como el de la figura 2.15. Vemos en esta figura que un mínimo local de gran magnitud puede atrapar al robot. Estos mínimos locales de gran magnitud suelen estar vinculados a la existencia de paredes, y para tratar con ellos se definirán más adelante otro tipo de mecanismos.

2.8 Generalización del método de las cargas ficticias

Aunque en principio se trata de un tema que se aparta un poco de los objetivos de esta tesis, surgió la cuestión de si el mecanismo de las cargas ficticias podría aplicarse en un ámbito más amplio, como es el de la exploración en espacios de estados. En efecto, en determinados algoritmos de este paradigma, como en el método de escalada, surge también el problema de la aparición de mínimos locales en la función heurística. De hecho el método del potencial es un caso particular de algoritmo de escalada aplicado al dominio del razonamiento espacial.

La detección de mínimos locales es también posible en un caso general; puesto que conocemos el estado objetivo, podemos determinar si estamos en un estado distinto del objetivo y en el que la función heurística se hace localmente mínima, es decir, aumenta en todas las transiciones posibles desde el estado actual.

En estas condiciones se puede generar una carga ficticia en la función heurística añadiendo un nuevo término en la misma. Siendo $h_0(x,o)$ la función heurística que estima la distancia entre los estados *x* y *o*, y sabiendo que en el estado *m* se ha detectado un mínimo local, podemos definir la nueva función heurística $h_{n+1}(x,o)$ como

$$h_{n+1}(x,o) = h_n(x,o) + \frac{\alpha}{h_n(x,m)} \qquad (2.16)$$

donde α es un parámetro que indica la intensidad de la carga repulsiva colocada. Puede observarse que, una vez fijada la carga ficticia, mientras más próximo se encuentre el estado *x* al estado *m* (el mínimo local), mayor será el valor que tome la nueva componente del heurístico, y por tanto los estados cercanos a *m* tenderán a no ser elegidos, lo que provocará que el sistema se aleje del estado *m* y sus estados próximos medidos según la función heurística previa a la modificación.

En la bibliografía aparecen dos versiones básicas del método de escalada [Rich & Knight, 1994; Winston, 1994]. En la primera de ellas no se guardan los caminos pendientes, de modo que el algoritmo se atasca si encuentra un mínimo local en la función heurística. En la segunda —la más habitual— es necesario almacenar en memoria los caminos pendientes de exploración, de modo que, si se detecta un mínimo local, el algoritmo pueda hacer *backtracking* retornando a un camino pendiente anterior. El almacenamiento de los caminos pendientes puede consumir una enorme cantidad de memoria en problemas complejos.

El algoritmo que presentamos no necesita almacenar los caminos pendientes ni hacer *backtracking*, y gracias a la modificación del heurístico no queda atrapado en los mínimos. El esquema del algoritmo de exploración basado en cargas ficticias generalizadas es el siguiente:

SEARCH-WITH-CHARGES (start end h)

1) Si el estado actual es el objetivo, terminar, hemos recorrido el camino hasta la solución.

2) Calcular los sucesores del estado actual y su valoración heurística. Si la valoración heurística de todos los sucesores es mayor que la del estado actual, estamos en un mínimo local: en este caso modificar el heurístico con la ecuación 2.16 e ir a 2.[4]
3) Seleccionar como estado actual el estado sucesor de menor valor heurístico. Ir a 1.

2.9 Extensión del método del potencial a robots no holonómicos y sin simetría sensorial

Aunque las ecuaciones planteadas en este capítulo responden al caso de robots holonómicos[5], la formulación original de la planificación de trayectorias con campos artificiales de potencial permite la especificación de robots con cinemática más compleja (de hecho se concibió para ser aplicada a manipuladores). El modelo propuesto en este capítulo puede extenderse fácilmente al caso de robots no holonómicos, sin más que modelizar al robot como un segmento y calculando las fuerzas a aplicar en sus extremos. Los extremos del segmento dependen del punto de aplicación de los sensores disponibles. En un robot con dos ruedas directrices delanteras puede ser una buena solución considerar el punto de aplicación de las fuerzas el eje delantero, aunque para obtener garantías de no colisión de la parte trasera sería preciso disponer de sensores y ruedas directrices traseras o bien añadir al sistema la capacidad de efectuar maniobras.

Una ventaja del uso del método del potencial es que la planificación de la navegación puede realizarse directamente en el espacio de operaciones. No es necesario transformar el espacio de operaciones en el espacio de configuración, transformación que precisa conocer completamente el entorno para ser llevada a cabo, lo que es un grave inconveniente para un sistema verdaderamente autónomo.

---

[4] Obsérvese que la modificación de heurístico puede realizarse repetidas veces, de modo que la aparición de varios mínimos locales iría acumulando términos en la expresión.

[5] Un robot holonómico es aquél cuya cinemática le permite acceder directamente (sin maniobras) a cualquier posición del espacio de operaciones.

Cuando el robot posee simetría sensorial, como es el caso del NOMAD 200 —utilizado en esta tesis—, las fuerzas de repulsión debidas a los obstáculos se calculan a partir de las medidas de los sensores aplicando la ecuación 2.2. Esta ecuación puede extenderse a sistemas sensoriales no simétricos considerando que cada sensor real (representados en gris en la figura 2.16) incorpora un sensor ficticio de orientación opuesta (representados en blanco) cuyo rango se fija permanentemente al alcance máximo de los sensores reales (ecuación 2.17). Utilizando esta técnica podemos garantizar que las fuerzas debidas a los sensores quedan compensadas en ausencia de obstáculos. En realidad esta propuesta es una generalización del caso del robot con sensores simétricos, en el cual las fuerzas de los sensores ficticios se anulan dos a dos, simplificándose el cálculo.

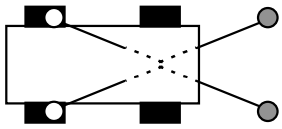


Figura 2.16. Un robot sin simetría sensorial

$$\vec{S} = \sum_i k_s \left(\frac{1}{rmax_i^{m_s}} - \frac{1}{r_i^{m_s}} \right) \vec{o}_i \qquad (2.17)$$

Un último comentario de generalización al caso de sensores binarios. Este tipo de sensores —por ejemplo los sensores táctiles— sólo dan valor 1 o 0, es decir, activado o desactivado, en lugar de un valor numérico entero o real. Si no deseamos que el valor de $\vec{S}$ cambie bruscamente podemos tomar en lugar de r_i el promedio de las últimas n medidas.

2.10 Resultados

En este capítulo hemos presentado un sistema de navegación basado en el método del potencial estimado en tiempo real a partir de la información suministrada por los sensores. Nuestras aportaciones a los métodos clásicos son las siguientes:

a) Hemos añadido una nueva característica al método: la posibilidad de añadir cargas ficticias con objeto de modificar el campo de potencial en las posiciones donde se ha detectado un mínimo local. Esta modificación hace que el robot se aleje del mínimo de un modo natural, sin necesidad de modificar, como proponen otros autores, la estrategia de navegación.

b) Se han generalizado las expresiones de estimación de la fuerza incidente sobre el robot para cualquier exponente. Dicho exponente regula la forma del módulo de la repulsión, de manera que es posible conseguir efectos a distancia variable.

c) Se ha proporcionado una expresión para ajustar los parámetros de repulsión *k* y *m* dadas las fuerzas repulsivas deseadas para dos distancias. Ello nos permite efectuar el ajuste de los parámetros de repulsión con comodidad.

d) Se ha generalizado el método de las cargas ficticias para su uso con algoritmos de escalada. Este método generalizado proporciona un mecanismo de exploración de espacios de estado enormemente simple y con un consumo de memoria despreciable que auto-modifica su función heurística para escapar de los mínimos locales.

e) El modelo de la estimación del potencial en tiempo real ha sido extendido para su uso en robots sin simetría sensorial.

f) Hemos presentado una serie de ejemplos de navegación obtenidos utilizando un simulador del robot. En el capítulo 8 presentaremos resultados experimentales de navegación utilizando el robot real.

3. Gradiente sensorial y Lugares Sensorialmente Relevantes

En este capítulo queremos sentar las bases teóricas que permitan la construcción —que se abordará en el capítulo siguiente— de un sistema cognitivo capaz de aprender la topología del entorno de forma que sea posible elaborar posteriormente planes tentativos globalmente óptimos. El problema que enfrentamos es ¿qué procesos perceptivos necesitamos definir para obtener información útil a los mecanismos de planificación de trayectorias? Intentaremos dar respuesta a este problema introduciendo algunos conceptos innovadores: el *gradiente sensorial* y los *lugares sensorialmente relevantes*.

Dada la complejidad y la cantidad de información que suministran los sensores, debemos previamente definir alguna estrategia que ayude a separar el grano de la paja, es decir, es necesario establecer mecanismos que, a modo de señaladores, identifiquen estados, experiencias o percepciones que deban ser recordadas para el futuro. Trabajos anteriores [Lozano-Pérez & Wesley, 1979; Kumpel & Serradilla, 1990; Serradilla, 1992] prueban que para obtener planes útiles no es necesario en absoluto recordar todos los puntos en los que estuvo previamente el robot; bastan unos pocos, distribuidos estratégicamente, para construir una estructura en forma de grafo que, encapsulando el conocimiento suficiente sobre el entorno del robot, permita afrontar la resolución de nuevos problemas.

3.1 El mundo es computacionalmente inabordable

El mundo real es continuo, o al menos inmensamente rico en información. Esta ingente cantidad de información plantea dos serios problemas a los sistemas inteligentes. El primero de ellos es que su almacenamiento "en crudo" es imposible. El segundo es que el razonamiento utilizando la manipulación directa de esta información es inabordable. El propio hombre reduce tal cantidad de información a modelos simples de la realidad, de forma que sean computacionalmente tratables e isomórficos con el mundo [Hofstadter, 1992]. De esta manera el ser humano puede (a) traducir una situación a uno de tales modelos, (b) tomar decisiones con objeto de alcanzar algún estado objetivo, y (c) aplicando estas decisiones al mundo real, acceder (si el modelo es correcto) al estado objetivo deseado. Obsérvese, por ejemplo, que una gran parte del trabajo desarrollado en las actividades educativas supone un aprendizaje de estos tres puntos aplicados a dominios de conocimiento concretos.

Por todo esto, una de las tareas que debería ser capaz de abordar un sistema de IA dotado de capacidades perceptivas es la de integrar su información sensorial en un modelo coherente y fácilmente utilizable para desenvolverse en el entorno. La mayor parte de los sistemas de IA parten, sin embargo, de una categorización previa realizada por el diseñador, que suele ser óptima para la resolución del problema. No obstante, es fundamental, si queremos construir sistemas autónomos, desarrollar técnicas que permitan que dicha categorización se realice de modo automático. Esta necesidad de independencia en la categorización ha sido reclamada como futuro campo de investigación por Shen [Shen, 1994].

Las propiedades básicas que debe cumplir un modelo son dos: que la necesidad de recursos del sistema sea baja y que sea suficientemente rico como para permitir que el isomorfismo con el mundo real sea lo más completo posible. La primera tiende a reducir el tamaño y la complejidad del modelo, mientras que la segunda tiende a incrementarlos. En un enfoque práctico nos bastará que los modelos se sitúen entre dos cotas teóricas. La cota inferior representa un tamaño de modelo por debajo del cual se pierde el isomorfismo con la realidad, y por tanto las inferencias del sistema no sirven. La cota superior representa un tamaño de modelo por encima del cual se exceden las capacidades de almacenamiento y/o el tiempo de cómputo (que depende drásticamente del tamaño del modelo) es demasiado elevado para que el sistema sea útil. Naturalmente que estas cotas dependen del tipo de trabajo a desarrollar.

Hemos argumentado la necesidad de discretizar o categorizar el mundo real pero, ¿cómo hacerlo? Existen algunos trabajos recientes [Mataric, 1992; Walker *et al*, 1993; Kurz, 1996], que tienen como objetivo establecer algún tipo de clasificación de los puntos por los que atraviesa el robot, normalmente utilizando aprendizaje no supervisado, y más concretamente mapas autoorganizados [Kohonen, 1982]. El sistema identifica el lugar del mundo en el que se encuentra utilizando la información suministrada por los sensores del robot como apoyo a la información odométrica. De este modo se divide el mundo en una serie de regiones sobre las que se pueden posteriormente planificar rutas.

Otros trabajos más refinados intentan aplicar algún tipo de transformación a la información sensorial y buscar máximos locales de los valores obtenidos al variar la posición del robot. Los lugares que maximizan los valores son lugares diferenciados (*distinctive places*) que se memorizarán en una estructura de grafo [Kuipers & Byun, 1988; Pierce & Kuipers, 1994]. Podríamos considerar nuestro trabajo dentro de esta última línea, aunque introduciendo un nuevo operador, el gradiente sensorial, en el espacio del problema. En la sección siguiente exponemos el concepto de gradiente sensorial como herramienta básica para detectar cambios bruscos en el entorno y distinguir hechos o situaciones que deben ser recordados.

3.2 El concepto de Gradiente Sensorial

La idea subyacente al uso del gradiente sensorial es que, mientras los cambios en el mundo son suaves, la probabilidad de aparición de callejones sin salida (mínimos locales) al resolver una tarea es pequeña. Para comprender esto hagamos un experimento mental: piénsese en una gran habitación vacía. Siempre que el objetivo se encuentre dentro de la misma, no existe ningún mínimo local; el robot puede dirigirse libremente hacia cualquier objetivo que especifiquemos. Añadamos ahora objetos o paredes en la habitación. A medida que añadimos elementos, la información sensorial se hace más compleja, y paralelamente comienzan a aparecer mínimos locales. Ya no es posible llegar al objetivo simplemente dirigiéndose en la dirección en que éste se encuentra.

En los casos de cambio suave pueden utilizarse estrategias de exploración muy simples (en general, métodos de escalada o descenso del gradiente), por ejemplo dando respuestas reactivas a la información sensorial captada del entorno. Sin embargo, cuando aparecen

cambios sensoriales bruscos inevitablemente surgen mínimos locales; entonces se hace necesaria la elaboración de un plan, debido a que una toma de decisiones reactiva nos conduciría fácilmente situaciones de mínimo local en el espacio del problema. Para intentar construir modelos útiles en la evitación de mínimos locales nos va a interesar por tanto detectar y recordar estos cambios bruscos. Obsérvese que un cambio brusco en la información sensorial equivale a la idea intuitiva de "sorpresa". De modo natural nuestra atención se centra en lugares donde aparece un destello, se producen movimientos rápidos o surge un sonido estridente, hechos que se explotan, sin ir más lejos, en los anuncios publicitarios. Habitualmente los estados donde se producen cambios de información suelen ser recordados mejor que las situaciones rutinarias, y ello unido a la información relevante para la tarea (en el caso de la publicidad, el nombre del producto).

En un pasaje revelador de Neisser [Neisser, 1976, pag. 109], referido a la percepción visual, se argumenta la importancia del cambio en relación con el desarrollo de modelos cognitivos: *"El movimiento cambia la información estimular disponible de muchos modos. Incluso un pequeño desplazamiento de la cabeza es suficiente para revelar nuevos aspectos de la mayoría de los objetos cercanos y de ocultar otros que eran visibles anteriormente. Los patrones de ocultación y desocultación producidos de este modo especifican las posiciones relativas del observador y de los propios objetos. Movimientos más amplios tienen consecuencias visuales más dramáticas. Cuando pasamos alrededor de una esquina o a través de una puerta, obtenemos vistas completamente nuevas que estaban previamente ocultas"*.

Dentro de las tareas de razonamiento espacial, se producen cambios bruscos siempre que efectuemos una transición de ámbito espacial, precisamente al atravesar una puerta o doblar una esquina. Son éstos lugares los que básicamente interesan a la hora de planificar rutas.

3.2.1 Espacio de sensaciones

Definiremos *espacio de sensaciones* como un espacio n-dimensional donde cada dimensión se corresponde con la medición de uno de los sensores de un sistema. Definido así, un registro completo de la entrada sensorial de un sistema (imagen sensorial) puede representarse como un punto en el espacio de sensaciones, y la evolución temporal de las sensaciones podrá

describirse como una trayectoria (ver figura 3.1). En este espacio podríamos aplicar técnicas de reconocimiento de formas para identificar diferentes tipos de "percepciones", establecer diferentes categorías y eventualmente asociarlas a diferentes decisiones o acciones a desarrollar por el robot.

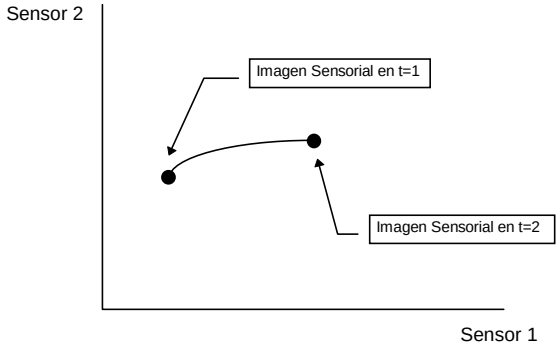


Figura 3.1. El espacio de sensaciones

3.2.2 Gradiente Sensorial

Definiremos *gradiente sensorial* como la derivada direccional de una trayectoria en el espacio de sensaciones. Este gradiente puede aproximarse como la diferencia entre los vectores que describen las entradas sensoriales en dos instantes sucesivos dividida por el periodo de muestreo (ecuación 3.1). Al espacio en el que representaremos los gradientes sensoriales lo llamaremos *espacio de gradiente sensorial*, y tendrá el mismo número de dimensiones que el espacio sensorial.

$$\nabla\vec{s} = \frac{\vec{s}(t + \Delta t) - \vec{s}(t)}{\Delta t} \qquad (3.1)$$

donde $\vec{s}(t)$ es la imagen sensorial en el instante t y Δt es el periodo de muestreo.

Cuando un sistema se mueve en el espacio, como es el caso de un robot móvil, el gradiente sensorial puede calcularse con respecto a la posición del robot, según la ecuación 3.2.

$$\nabla\vec{s} = \frac{\vec{s}(\vec{p}_n) - \vec{s}(\vec{p}_{n-1})}{\left|\vec{p}_n - \vec{p}_{n-1}\right|} \qquad (3.2)$$

donde $\vec{s}(\vec{p}_n)$ es la imagen sensorial en la posición $\vec{p}_n$.

En muchos sistemas sensoriales, como los que utilizan ultrasonidos, el gradiente sensorial es extremadamente sencillo de calcular; con otro tipo de sensores el cálculo puede ser más complejo. No obstante la idea de gradiente sensorial tiene una amplia generalidad, pudiendo aplicarse, entre otras, a información visual, sensores de luminosidad, infrarrojos, sonido, etc. En lo que sigue nos limitaremos a su aplicación a sensores de ultrasonidos e infrarrojos en tareas de razonamiento espacial, aunque al final del capítulo comentaremos someramente la posible utilización de cámaras de visión artificial.

Sobre el espacio sensorial podemos aplicar todas las operaciones tradicionales de la teoría de análisis de señales: filtrados, transformada de Fourier, etc., y las de la teoría de reconocimiento de formas. En particular, el operador gradiente nos proporciona un mecanismo matemático para la determinación de la intensidad del cambio producido entre dos lugares próximos, ya que las percepciones del sistema sensorial han de ser muy diferentes al cambiar de habitación o al pasar junto a una esquina. Por el contrario, cuando las percepciones cambien suavemente, estaremos en una situación en la que no hay novedades importantes. Denominaremos *lugar sensorialmente relevante* (LSR) a una situación en la que el cambio sensorial con respecto al instante anterior es suficientemente alto. Cada vez que el sistema identifique un LSR deberá asimilarlo y relacionarlo con otros LSRs ya conocidos para establecer un modelo sobre el que posteriormente poder planificar misiones. Para detectar cuándo el robot se encuentra en un LSR a partir del operador gradiente sensorial, proponemos —entre otras— las siguientes estrategias:

1. Utilizar el módulo del gradiente. Si el módulo está por encima de cierto umbral, estamos en un LSR.
2. Utilizar mecanismos de aprendizaje no supervisado, partiendo de la suposición de que un LSR es por propia naturaleza diferente de otros lugares, ya que el gradiente sensorial toma en ellos valores distintos a los del resto. En particular, podríamos utilizar un mapa autoorganizado [Kohonen, 1982; Hertz *et al*, 1991] que aprendiera a detectar LSRs a partir del gradiente sensorial. Este uso de los mapas autoorganizados difiere de otros propuestos en la literatura en que, en lugar de utilizar la información sensorial “en crudo”, utiliza como entrada a la red el resultado de aplicar el gradiente sensorial.

3. Utilizar mecanismos de aprendizaje supervisado, que instruyan al sistema sobre cuándo estamos en un LSR. Estos tienen el inconveniente de necesitar un maestro.

Hemos desestimado la estrategia 3 debido a que no es un procedimiento completamente autónomo, aunque puede ser interesante estudiarla en el futuro. Exponemos a continuación las estrategias 1 y 2.

3.2.3 Detección de LSRs utilizando el módulo del gradiente sensorial

Para determinar cuándo un lugar pertenece a la categoría de los LSRs podemos utilizar la ecuación 3.3.

$$\vec{p} \in LSR \Leftrightarrow |\nabla \vec{s}| > U \qquad (3.3)$$

donde $\vec{p}$ es la posición actual del robot, $\nabla \vec{s}$ se refiere al valor del gradiente sensorial (que puede calcularse con las ecuaciones 3.1 o 3.2) y U es un valor umbral.

En la figura 3.2 se muestra el resultado de calcular el módulo del gradiente sensorial en todos los puntos de un mundo conocido. La intensidad de los *pixels* es proporcional a la magnitud del módulo; no se ha realizado ninguna umbralización. Los puntos en blanco corresponden a posiciones de cambio suave y los puntos oscuros a posiciones de cambio brusco. El cálculo se ha realizado situando al robot simulado en la posición (x,y) y tomando una imagen sensorial. Seguidamente se ha desplazado el robot a la posición $(x+1, y+1)$[6] y se ha vuelto a tomar su imagen sensorial. Con ambas imágenes hemos calculado el gradiente y el módulo del vector resultante. En el proceso hemos evitado posiciones imposibles para el robot, tales como las que interceptan una pared. En la figura 3.3 se muestra un detalle de una zona de la figura 3.2.

[6] Teóricamente, el robot podría desplazarse a cualquier punto colindante con el punto (x,y). El hecho de utilizar el punto (x+1,y+1) es una simplificación a efectos de poder generar la imagen del ejemplo. En cualquier caso, si se utilizan otros puntos colindantes el resultado es similar.

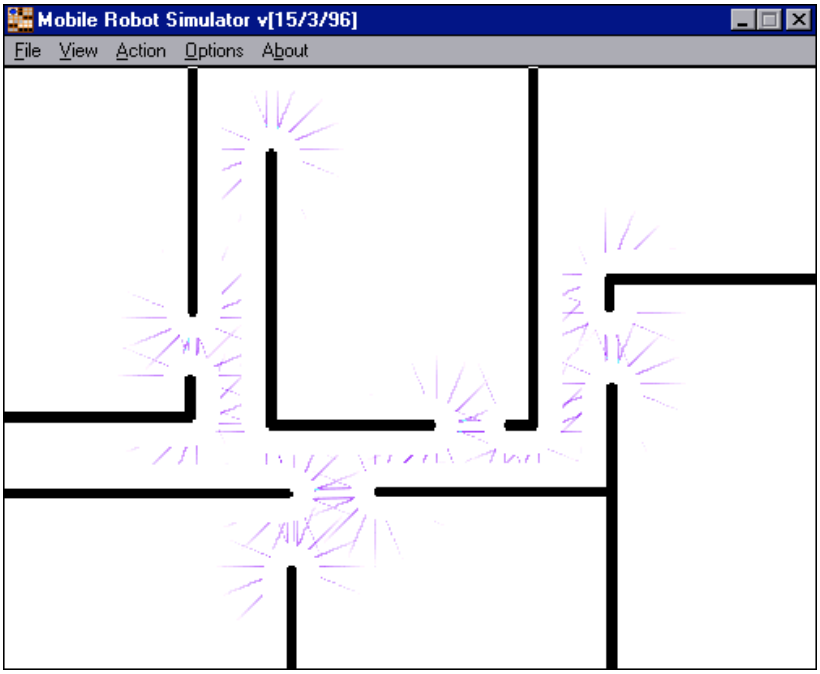


Figura 3.2. Puntos con alto gradiente en un mundo simulado

Una variante al cálculo del módulo que genera un número menor de LSRs consiste en utilizar sólo algunas de las componentes del vector gradiente, en especial las que corresponden a zonas laterales del robot.

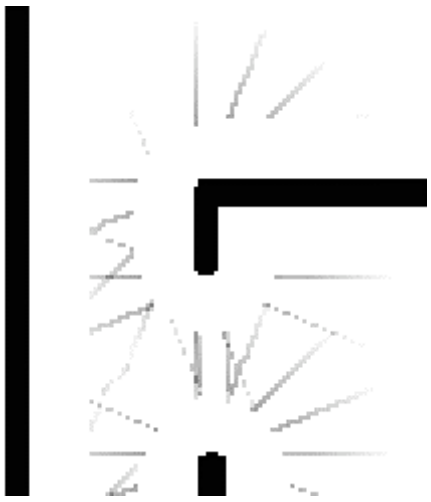


Figura 3.3. Puntos con alto gradiente en un mundo simulado

El problema que plantea la determinación de LSRs a través del módulo del gradiente es la elección del umbral *U*. La distribución de los valores del módulo del gradiente varía según el número de sensores del sistema y según el alcance mínimo y máximo de los mismos.

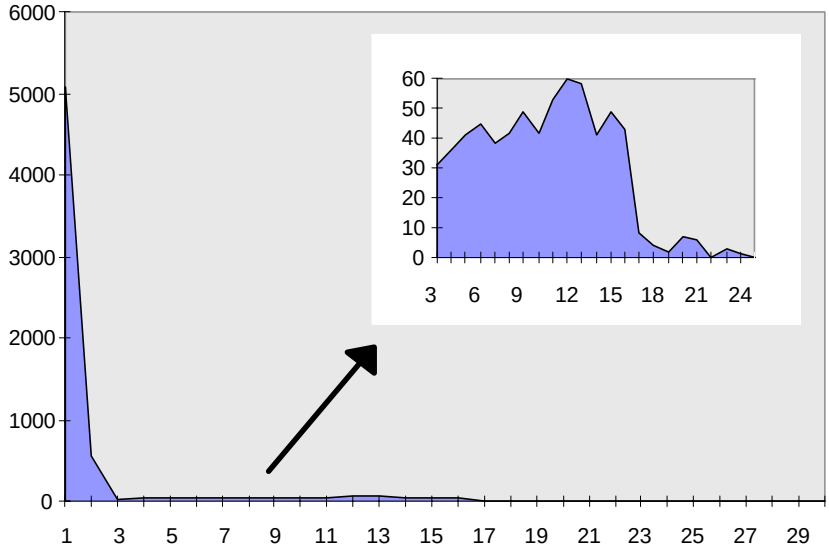


Figura 3.4. Histograma de la distribución del módulo gradiente sensorial. El cuadro apuntado por la flecha es una ampliación de la zona correspondiente.

La figura 3.4 muestra un histograma de la distribución que toman los valores del módulo del gradiente sensorial durante una misión simulada (en el capítulo 8 mostraremos datos de misiones reales). Puede apreciarse que el número de casos en los que el módulo del gradiente es mayor que 3 es extremadamente bajo. Esto coincide con nuestras expectativas, ya que los lugares de cambio brusco deben ser escasos, predominando los lugares en los que "no pasa nada". Podemos fijar el umbral en algún sitio por encima de 3. Mientras más alto sea el umbral, menor será la probabilidad de detectar un LSR; mientras menor sea el umbral, más LSR serán detectados. El umbral ideal dejaría dentro de la zona de detección todas las puertas y esquinas, excluyendo los lugares de variación debidos al ruido, pequeños objetos, etc.

Hemos determinado que para sensores de ultrasonidos puede obtenerse un umbral adecuado aplicando la ecuación 3.4, donde r_{min} y r_{max} representan, respectivamente, el alcance mínimo y máximo de los sensores. En esta ecuación puede observarse que el número de sensores no afecta al umbral de detección. Esto parece lógico ya que raramente cambian bruscamente las medidas de dos sensores al mismo tiempo. Si sólo cambia un sensor, asumiremos que un cambio es significativo cuando supere entre el 15 y el 20% de la diferencia entre alcance máximo y alcance mínimo.

$$U = m \cdot (r_{max} - r_{min}) \qquad (3.4)$$

con $m \in [0{,}15,\ 0{,}2]$.

En el caso del simulador, r_{min} = 6 y r_{max} = 70, de modo que esta ecuación determina un valor de $U \in$ [9,6, 12,8]. En el caso del robot NOMAD-200, con r_{min} = 6 y r_{max} = 255, obtenemos un valor de $U \in$ [37,35, 49,8].[7]

Otra opción consiste en usar un umbral adaptativo que garantice obtener el porcentaje de LSRs que deseemos. En la ecuación 3.5 proponemos una expresión general para la determinación del umbral de detección a partir del porcentaje de LSRs que se desea detectar. Dicho porcentaje equivale al área del histograma que queda a la izquierda del umbral. Este valor se puede recalcular en cada iteración para adaptarlo a las condiciones del entorno de trabajo.

$$U = k \text{ tal que } \sum_{i=1}^{k} H(i) = (1 - p_d) \sum_{i=1}^{N} H(i) \qquad (3.5)$$

donde $H(i)$ hace referencia a la entrada i del histograma, N es el número total de entradas del histograma y p_d es la probabilidad deseada de detección de LSR. Al estar el histograma compuesto de valores discretos, la igualdad no tiene por qué verificarse exactamente: bastará que elijamos el valor de k que haga más parecidos a los dos términos.

En caso de duda es preferible el exceso a la carencia de LSRs. El exceso de LSRs aumenta la carga computacional, ya que habrá que comprobar más veces si el LSR detectado ya existe en el modelo[8], pero la carencia puede redundar en un mal funcionamiento del sistema, debido a que los LSR estarán muy alejados unos de otros y será difícil que el navegador pueda completar sus misiones de enlace entre ellos.

3.2.4 Detección de LSRs utilizando mapas autoorganizados

Una alternativa interesante es usar como detector un mapa autoorganizado o red de Kohonen. Un mapa autoorganizado es básicamente un clasificador no supervisado cuyas clases están

[7] El alcance proporcionado por los sensores puede recortarse a un valor menor que su alcance máximo para evitar la imprecisión del los ultrasonidos a largas distancias. Por ejemplo, recortando r_{max} a 100, $U \in$ [14,1, 18,8].

[8] Como posteriormente veremos, un LSR sólo se creará si en las inmediaciones no existe ya uno, de modo que no es especialmente problemático tomar un umbral bajo.

dispuestas en una retícula de neuronas conocida como la capa competitiva. La característica más relevante de los mapas autoorganizados es que preservan la topología de los datos, es decir, objetos cercanos en el espacio del problema activan neuronas cercanas en la capa competitiva. Para más detalles sobre el entrenamiento y propagación de señales en los mapas autoorganizados consultar [Kohonen, 1982; Freeman, 1991; Hertz *et al*, 1991].

Este tipo de clasificador ha sido redescubierto por los investigadores en robótica móvil [Mataric, 1992; Walker *et al*, 1993; Kurz, 1993; 1996; Keymeulen & Decuyper, 1992], quienes lo han utilizado para identificar regiones del entorno y de este modo estimar la localización del robot en combinación con la información odométrica. En nuestro trabajo hemos utilizado los mapas autoorganizados para resolver el problema de la detección de LSRs a partir del gradiente sensorial.

La hipótesis de trabajo es que, suministrando valores de gradiente sensorial como entrada a la red, ésta debe ser capaz de aprender al menos dos categorías: los gradientes que corresponden a lugares "irrelevantes" y los gradientes que corresponden a LSRs. Además, aprovechando que las redes de Kohonen preservan la topología, podemos utilizar una capa competitiva de 1×N para obtener una gradación de situaciones, en el centro los casos "irrelevantes" y en los extremos los LSRs más "severos", y posteriormente etiquetar las neuronas que nos interesa que sirvan como detectores de LSRs.

Como el equipo sensorial del robot es simétrico, podemos generar ejemplos virtuales o hipotéticos [Abu-Mostafa, 1994; 1995] que incrementarán la capacidad de generalización de la red.

La figura 3.5 muestra los prototipos de las clases obtenidas tras el entrenamiento de una red de Kohonen de 16 entradas (una por sensor) y 5 neuronas en la capa competitiva. La figura 3.6 muestra los prototipos de las clases obtenidas con 16 entradas y 16 neuronas en la capa competitiva. El entrenamiento se realizó sobre un conjunto de 50399 ejemplos tomados durante una misión del robot. En las figuras se representa el valor final de las conexiones entre cada entrada sensorial E_i y cada neurona. Un nuevo registro sensorial activará la neurona cuyas conexiones tomen valores más cercanos a dicho registro. La activación de cada neurona implica el reconocimiento de una clase o categoría diferente. Junto a cada clase hemos

anotado el número de ejemplos del conjunto de entrenamiento que activan la neurona correspondiente.

Analizando los resultados de la figura 3.5 podemos observar que la neurona 3 (representada en la fila central) se ha especializado en detectar lugares "irrelevantes", mientras que el resto de neuronas se han especializado en detectar cierto tipo de gradiente. Así, por ejemplo, la neurona 1 (representada en la primera fila) se ha especializado en detectar cambios negativos en el sensor 13.

Comentaremos dos hechos. El primero es que la mayoría de los ejemplos (49408 del total, 50399) corresponden a lugares "irrelevantes". Esto supone un 98% de los casos, lo cual está de acuerdo con la necesidad de que el número de LSRs sea bajo. El segundo es que el clasificador se especializa en observar determinados sensores. Ello parece indicar que la reducción de dimensionalidad obliga a elegir determinados sensores antes que otros.

El entrenamiento se ha realizado repetidas veces con resultados similares.

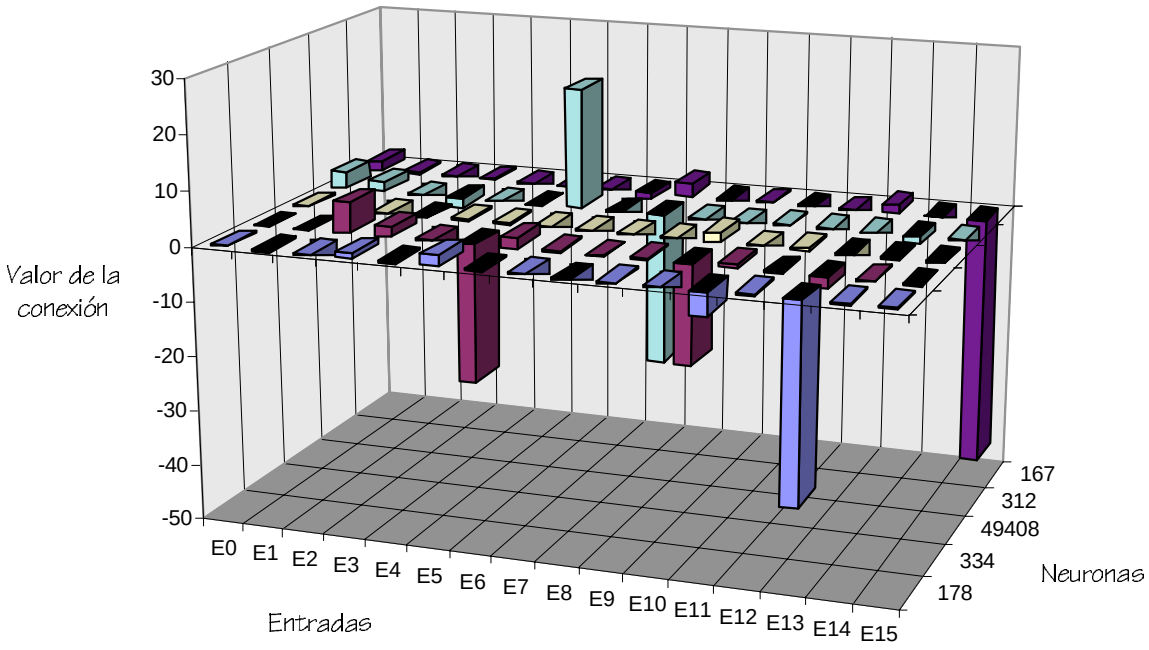


Figura 3.5. Prototipos de las clases aprendidas por el mapa autoorganizado con una capa competitiva de 1x5

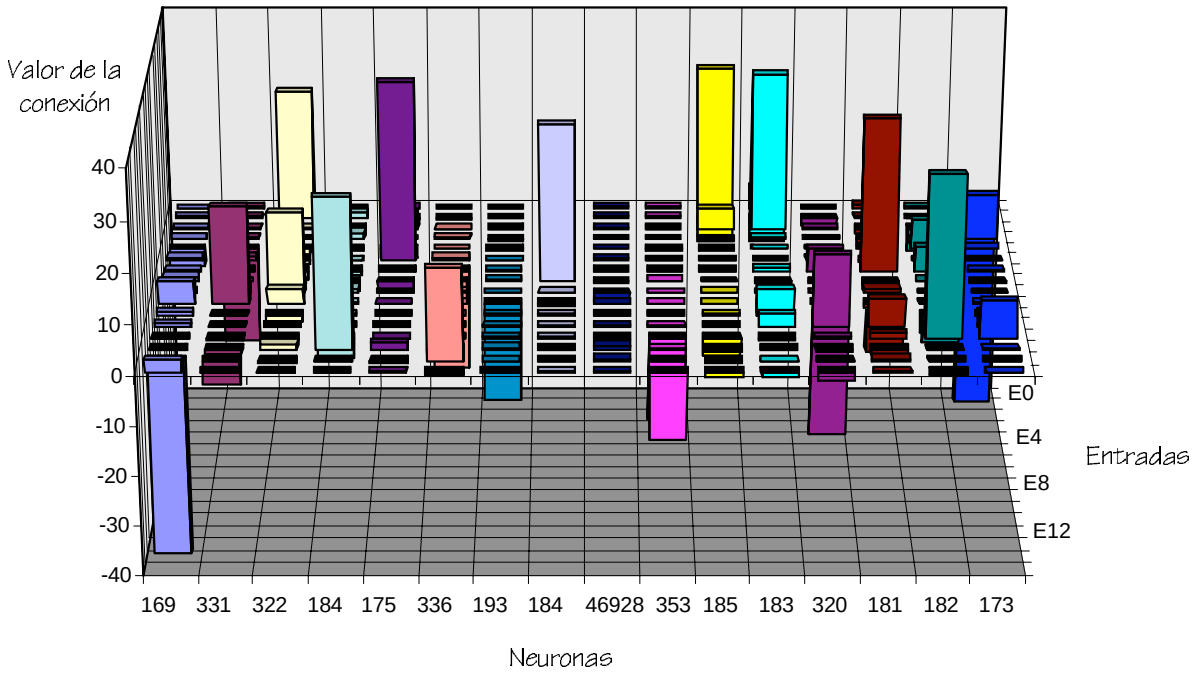


Figura 3.6. Prototipos de las clases aprendidas por el mapa autoorganizado con una capa competitiva de 1x16

Los resultados de la figura 3.6 muestran una mayor diversidad de clases; no obstante, el grueso de los ejemplos sigue correspondiendo a los lugares "irrelevantes" (clase 9, con 46928 ejemplos, es decir, el 93% del total). Como puede observarse, en ambos casos se ha especializado alguna neurona (curiosamente en la posición central) en detectar gradientes cuyas componentes son prácticamente cero. Además, todos los sensores participan en mayor o menor medida en la detección de alguna clase.

Las pruebas de detección realizadas con mapas autoorganizados arrojan resultados positivos: los LSRs detectados son suficientes para construir un mapa completo. Sin embargo, debido a que las clases se especializan en algunos tipos de gradiente, hay algunas situaciones de alto cambio que no son identificadas como LSR. La consecuencia última de este problema es que la detección no es simétrica: se hace dependiente de la dirección de aproximación del robot al LSR, es decir, del sensor con que se mira. Esto es un efecto no deseado porque nos obligaría a renunciar a una concepción no-dirigida del modelo. Como conclusión, el módulo es preferible a los mapas autoorganizados, si bien estos últimos dan cierto fundamento "natural" al gradiente sensorial y corroboran su utilidad como detector de *landmarks*.

3.3 Propiedades de los LSRs como puntos de referencia

Los seres humanos, cuando planificamos una ruta en un entorno complejo, subdividimos el plan en pequeñas partes, tomando puntos de referencia que podemos localizar con nuestros sensores (en el caso del hombre mediante la visión, fundamentalmente). Estos puntos de referencia juegan un papel clave en los procesos de planificación, y deben poseer las siguientes características:

- Ser fácilmente identificables por el sistema sensorial.
- Su número debe ser pequeño para que el modelo ocupe el menor espacio de memoria posible y la planificación sobre ellos sea eficiente.
- Su número debe ser suficientemente alto como para que no existan mínimos locales entre dos puntos de referencia directamente conectados. Esto evitará que el sistema de navegación pueda quedar atrapado en un callejón sin salida. Una restricción menos severa es permitir la existencia de mínimos locales siempre y cuando éstos sean de magnitud suficientemente pequeña como para que un sistema de navegación de tipo reactivo los evite.
- Su posición no debe describirse en coordenadas absolutas, sino en coordenadas aproximadas relativas a otros puntos de referencia. Es decir, hablamos de modelos topológicos, no geométricos.

En el campo del razonamiento espacial proponemos utilizar los LSRs obtenidos a partir del gradiente sensorial como puntos de referencia para la planificación. En combinación con la detección de mínimos locales veremos en los capítulos siguientes que es posible, partiendo del concepto de LSR, construir automáticamente estructuras cognitivas que permitan una planificación de caminos eficiente.

En nuestro sistema, un LSR contendrá la siguiente información:

- Sus coordenadas espaciales aproximadas. Una vez detectado un LSR nos ayudarán a identificarlo y eventualmente nos servirán para actualizar las coordenadas del robot.
- Un mapa sensorial promedio. Este mapa sensorial contiene la imagen sensorial promedio captada desde el LSR, que se utilizará, junto con sus coordenadas

aproximadas, para identificar el LSR, así como para determinar direcciones válidas de exploración.

- Un vector con datos de cuadrantes practicables, de los que nos ocuparemos en el capítulo siguiente.
- Una lista con todos los LSRs accesibles desde él.

3.4 Cálculo del gradiente sensorial en visión artificial

Una aproximación al gradiente temporal entre dos imágenes puede obtenerse a través de la diferencia de imágenes [Sonka *et al*, 1993]. Habitualmente, dicho gradiente temporal se usa para determinar las regiones de la imagen que han sufrido modificaciones, por ejemplo como ayuda a la segmentación de un objeto. Como estimación absoluta del gradiente entre dos imágenes, es decir, para determinar si éstas han cambiado mucho o poco, proponemos la utilización de un sencillo *ratio*, expresado en la ecuación 3.6. En ella se calcula el nivel de gris medio de la imagen diferencia, de modo que, si este nivel medio es alto, podremos concluir que las diferencias entre las imágenes en los instantes *t* y *t-1* son grandes. Como siempre, habrá que fijar un umbral para determinar qué intensidad de gradiente debe dar como resultado la identificación de un LSR.

$$\nabla[I_t, I_{t-1}] = 1/N^2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} abs(I_t(i,j) - I_{t-1}(i,j)) \qquad (3.6)$$

donde I_t e I_{t-1} se refieren a la imagen tomada en los instantes *t* y *t-1*, respectivamente, y *abs* es la función valor absoluto, que podría sustituirse por la función cuadrática, que, aunque computacionalmente es más costosa de calcular, puede presentar ciertas ventajas como es la derivabilidad.

3.5 Reflexiones sobre las características deseables en el equipo sensorial

Las técnicas propuestas a lo largo de este capítulo son suficientemente generales como para ser utilizadas en una amplia variedad de equipos sensoriales. En particular pueden aplicarse directamente en sistemas sensoriales basados en medidas de rango: ultrasonidos, infrarrojos, telémetro láser y visión estereoscópica. En otros casos, como la visión monocular, hemos

visto que es posible obtener alguna estimación del gradiente sobre la que aplicar las técnicas de detección estudiadas.

Una característica deseable —aunque no imprescindible— es que el equipo sensorial disponga de simetría. Este hecho se podrá explotar a través de la construcción de modelos no-dirigidos en el espacio. Sin esta característica las técnicas propuestas continúan siendo igualmente válidas, aunque el aprendizaje del entorno será algo más laborioso porque la detección de un LSR dependerá de la dirección de aproximación del robot.

En resumen, simetría sensorial permite modelos no-dirigidos, mientras que asimetría sensorial obliga a la construcción de modelos direccionales.

Un comentario particular para el problema de detección de LSRs en entornos estructurados: hemos observado que en los edificios de oficinas, viviendas particulares, hospitales, etc., sería extremadamente simple detectar LSRs incorporando al robot un sensor ultrasónico que apuntase *hacia el techo*. Siempre que el robot atraviese una puerta, el marco de la misma producirá un cambio brusco en el rango medido. Con esta incorporación *ad hoc* bastará monitorizar el gradiente de dicho sensor cenital con alguna de las técnicas propuestas en este capítulo.

3.6 Propuesta de generalización del concepto de gradiente sensorial

En este capítulo hemos planteado la idea de gradiente sensorial y hemos defendido su utilidad como herramienta de discretización del mundo en estados útiles para establecer planes en un dominio concreto: el del razonamiento espacial. Sin embargo, pensamos que el concepto de gradiente sensorial es una propuesta de mucha más generalidad, y presumiblemente de utilidad en dominios muy dispares; siempre que dispongamos de una serie de medidas que cambian con el tiempo puede ser interesante analizar su variación y establecer *puntos de control* (una generalización de nuestros LSRs) en los lugares de cambio brusco de estas medidas.

El objetivo sería encontrar algún tipo de medida que nos garantice, fundamentalmente, que entre dos puntos de control cualesquiera la función que estima la distancia entre estados (heurística) sea monótona decreciente, y que, por ello, las estrategias basadas en el descenso

del gradiente (también conocidas como métodos de escalada) podrán utilizarse sin riesgo de quedar atrapados en un mínimo local. En robótica móvil esta medida puede ser el gradiente sensorial.

En campos de aplicación tales como la exploración en espacios de estados o la planificación de tareas, podemos sustituir la información sensorial por alguna medida alternativa asociada al problema. A modo de ejemplo considérese como medida el grado de conectividad de un nodo en un espacio de estados. Si la conectividad de un conjunto de nodos es alta, seguramente existen numerosos caminos entre ellos, y consecuentemente la probabilidad de caer en un mínimo local será pequeña. Si la conectividad es muy baja, lo más probable es que exista un único camino entre dos nodos, y el daño que pueda causar un mínimo local será mucho mayor. Cuando se pasa de una zona de alta conectividad a una de baja (o viceversa) el gradiente de la conectividad cambia bruscamente, que es exactamente lo que sucede en robótica móvil al pasar de una habitación a un pasillo (o viceversa). En un espacio de estados puede sernos útil detectar estos pasillos “virtuales” y recordar dónde se localizan sus entradas y sus salidas, los puntos de control.

En este contexto, los puntos de control constituirían meta-estados dentro de un cierto meta-espacio. Todos los estados situados entre dos meta-estados pueden ser ignorados puesto que desde un meta-estado a otro colindante en el meta-espacio podemos trasladarnos aplicando simplemente un criterio de descenso en la función de distancia, es decir, no es necesario un algoritmo de exploración de estados en sentido estricto. Bastaría, entonces, establecer una planificación en el meta-espacio (lo cual puede significar una reducción importante en la complejidad del problema) y posteriormente encontrar subplanes entre cada par de meta-estados aplicando un simple método de escalada.

Obviamente, analizar en detalle la viabilidad de esta propuesta requeriría el desarrollo de una tesis específica dentro del paradigma de la planificación de tareas. La conectividad, enunciada a modo de ejemplo, no parece aportar suficiente información como para detectar estos pasillos virtuales. Cuestiones abiertas son: ¿cómo puede completarse la conectividad con otro tipo de medidas? ¿Qué es lo que hace en los humanos que elijamos ciertos estados de un problema como subobjetivos? ¿En qué modo la información sensorial influye en esta elección? Hemos dado una respuesta a este problema en el campo del razonamiento espacial,

y pensamos que la idea de gradiente sensorial generalizado podría dar respuestas mucho más amplias en el futuro.

3.7 Resultados

En este capítulo hemos presentado el concepto de gradiente sensorial como una potente herramienta de discretización de la continuidad del mundo real. Basándonos en el gradiente sensorial hemos introducido la idea de LSR, defendiendo su importancia como punto de referencia para la planificación de rutas.

Así mismo, hemos desarrollado dos procedimientos de detección de LSRs: la detección basada en el módulo y la detección basada en mapas autoorganizados. Dejaremos para el capítulo siguiente el análisis de los resultados obtenidos sobre el uso de los métodos de detección de LSRs para la construcción del modelo cognitivo, dada la estrecha vinculación existente entre ambos procesos.

Para afinar la detección basada en el módulo del gradiente hemos propuesto dos expresiones que nos ayudan a estimar el umbral óptimo de detección, partiendo de las características del equipo sensorial o del histograma de distribución de gradientes. En cuanto a la detección basada en mapas autoorganizados hemos mostrado los resultados obtenidos mediante el entrenamiento, comparándolos con el procedimiento del módulo.

Se ha caracterizado el tipo de información que debe almacenar un LSR para su uso posterior como herramienta básica de construcción de modelos. Uno de los componentes más importantes de esta caracterización —el concepto de cuadrante de exploración— se expondrá en el capítulo siguiente.

Finalmente, hemos enunciado un método de estimación del gradiente para visión monocular, y hemos argumentado la posible generalización del concepto de gradiente sensorial a dominios de aplicación más amplios.

4. Nivel cognitivo. Modelado del entorno.

En este capítulo nos ocuparemos de cómo podemos utilizar las ideas previas —el gradiente sensorial, los lugares sensorialmente relevantes (LSR), la detección de mínimos locales y la navegación reactiva basada en el método del potencial artificial con cargas ficticias— para construir, mantener y utilizar un modelo topológico que intente reflejar la estructura del mundo con respecto a una tarea concreta: la planificación de rutas para un robot móvil.

Las primeras investigaciones en robótica móvil utilizaron modelos métricos del entorno: grafos de visibilidad [Nilsson, 1969; Lozano-Pérez & Wesley, 1979; Lozano-Pérez, 1987; Kumpel & Serradilla, 1989; Serradilla & Kumpel, 1989], diagramas de Voronoi [Ó'Dúnlaing & Yap, 1982], regiones libres [Brooks, 1983], etc. Dentro de esta línea clásica cabe citar como enfoque muy utilizado en la actualidad los métodos basados en rejillas (*grids*). Este sistema de basa en compartimentar el espacio en casillas cuadradas en las que se almacena una probabilidad de ocupación [Chatila, 1982; Elfes, 1989; 1990; 1991; Basye *et al*, 1989]. Una gran ventaja de estos métodos es la sencillez para integrar información proveniente de diferentes sensores; los inconvenientes son la necesidad de una gran exactitud en la determinación de la posición del robot y el alto coste computacional que conlleva la búsqueda de rutas a través de la rejilla, así como algunos problemas menores de navegación a través de las rutas obtenidas. De ellos, el más grave es el del elevado coste de la búsqueda, lo cual nos obligó —tras algunas pruebas previas— a desestimarlo para esta tesis.

En oposición a los modelos métricos encontramos los modelos topológicos. En ellos, la información fundamental no es la posición geométrica exacta de los objetos o *landmarks*, sino

su posición relativa a otros objetos. Algunos enfoques interesantes de modelado topológico pueden encontrarse en [Kuipers & Byun, 1988; Pierce & Kuipers, 1994; Mataric, 1992; Kurz, 1996].

Otra cuestión polémica surge al plantearnos en qué sentido el modelo debe reflejar el mundo. Las aproximaciones clásicas tenían la pretensión de identificar los objetos y sus posiciones para posteriormente —mediante razonamientos de carácter geométrico— establecer caminos libres de colisión. El problema es que la identificación de objetos es una cuestión compleja que hace inviable la operación en tiempo real.

Como se comentó en la introducción, algunos investigadores en psicología sostienen que la percepción es un proceso altamente orientado a la tarea [Gibson, 1977]. Desde este punto de vista, para la planificación de caminos no va a ser necesario extraer del entorno más información que la necesaria para la propia tarea de planificación, y los modelos construidos integrarán sólo esa información, lo que abarata considerablemente el coste computacional tanto de la extracción como de la construcción y mantenimiento del modelo.

De acuerdo con las teorías expuestas, en nuestro trabajo utilizamos dos suposiciones centrales:

a) Es necesario tener un modelo. Sin embargo es imprescindible que el sistema disponga de comportamientos reactivos que atiendan a las tareas más elementales, como la de evitar colisiones con los objetos. Asumimos, por tanto, un enfoque híbrido.

b) El modelo no debe ser geométrico ni reflejar en detalle el mundo, sino que estará orientado a la tarea. Será isomórfico con el mundo sólo en el aspecto que nos interesa.

4.1 Necesidad de un modelado cognitivo

Actualmente, en robótica móvil, existen dos posturas encontradas. La primera parte de una afirmación de Simon [Simon, 1969] en la que expresa que la complejidad del comportamiento de los seres vivos obedece más a la complejidad del entorno en el que se desenvuelven que a la propia complejidad de los organismos. Este punto de vista da origen a una corriente de

investigadores —muy vinculada con la robótica reactiva y con la vida artificial— que se plantean cómo explotar la complejidad del entorno en sistemas sencillos y no obstante capaces de desarrollar comportamientos complejos, *sin necesidad de elaborar representaciones internas explícitas* [Brooks, 1991].

La segunda, más en relación con la psicología cognitiva, postula que es imprescindible el desarrollo de modelos. Este desarrollo de modelos parece claro en el ser humano. Por ejemplo, hemos tomado de [Lindsay & Norman, 1983, pag. 581] el siguiente fragmento: *"Entre los 2 y los 7 años, el niño empieza a utilizar una representación interna de su mundo exterior. Este es su primer paso importante hacia el pensamiento adulto: la realización de experimentos mentales. Ahora, por vez primera, el niño puede predecir el curso de un suceso sin tener que producirlo en la realidad. Puede contestar a la pregunta siguiente: ¿qué sucedería si…?"*.

En esta tesis nos hemos adherido a esta segunda opción, de manera que gran parte de nuestro trabajo, resumido en los capítulos 3, 4 y 5, consiste en la búsqueda de mecanismos de construcción y utilización de modelos cognitivos.

4.2 Mínimos locales severos

En el capítulo 2 describimos un subsistema de navegación basado en el método del potencial artificial con la incorporación de la capacidad de añadir cargas ficticias. Se demostró que este método era capaz por sí mismo de eludir mínimos locales de cierta magnitud gracias a la auto-modificación de la función de potencial mediante la adición de cargas ficticias. No obstante, hay situaciones en las que este sistema no basta para encontrar un camino hasta un objetivo prefijado, o aún en caso de encontrarlo, la ruta seguida es poco eficiente. En la figura 2.13 vimos uno de tales ejemplos en los que el navegador queda atrapado en lo que hemos denominado un *mínimo local severo* (MLS, en lo sucesivo).

Un MLS es un mínimo local de magnitud suficiente como para que el método de las cargas ficticias (o, en su caso, el navegador reactivo utilizado) sea incapaz de salir de él. Vimos que esta situación depende del número máximo de cargas permitidas y de su tamaño, y puede identificarse porque *todas* las cargas permitidas han sido utilizadas en una región

suficientemente pequeña (ecuación 4.1). Si esto sucede sabemos que el robot está detectando una y otra vez la existencia de un mínimo aproximadamente en el mismo lugar, y por tanto podemos activar un *flag* que nos avise de ello.

$$d(\vec{m}_f, \vec{m}_0) < U \qquad (4.1)$$

donde $\vec{m}_f$ es la posición del último mínimo local detectado, $\vec{m}_0$ es el mínimo local más antiguo de los memorizados, *d* es la función distancia euclídea y *U* es un umbral. En nuestro trabajo, dicho umbral, ajustado experimentalmente, toma un valor de 60 pulgadas.

Resaltamos el hecho de que estos MLSs tan sólo aparecen en entornos lo suficientemente complejos. Hemos comprobado que en la mayoría de los ejemplos presentados en la literatura especializada no existen tales MLSs, y consecuentemente el método de las cargas es suficiente para encontrar una trayectoria satisfactoria cualesquiera que sean las posiciones inicial y objetivo. No obstante, es una de las ideas centrales de esta tesis el que un método puramente reactivo no basta para trasladarse con eficiencia en un entorno complejo con objetivos descritos posicionalmente, ya que en ellos sí puede aparecer este problema.

4.3 Cuadrantes de exploración

Para simplificar las posibles direcciones de exploración desde un LSR definiremos el concepto de *cuadrante de exploración*. Denominamos cuadrante de exploración a cada una de las *Q* regiones iguales en que podemos subdividir el continuo de las posibles orientaciones del robot, $\mathbf{o} \in [0, 2\pi)$. Así como la noción de LSR sirve para discretizar las posiciones útiles del entorno, el concepto de cuadrante discretiza la idea de dirección de movimiento. Mientras menor sea *Q* más eficiente será el sistema, pero menos preciso. En la figura 3 podemos ver la subdivisión en cuadrantes para $Q = 8$.

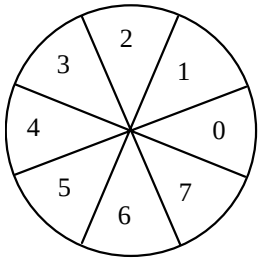


Figura 4.1. Subdivisión en cuadrantes de las direcciones de movimiento con Q = 8

Con la idea de disponer de datos sobre posibles direcciones de exploración, para cada cuadrante se almacenará un indicador que podrá señalar una entre tres posibilidades: (a) el cuadrante no ha sido explorado (q_i=0), (b) el cuadrante conduce a un MLS o a una pared

($q_i<0$) y (c) el cuadrante conduce a otro LSR ($q_i>0$). Denominaremos a estas tres categorías *cuadrantes no explorados*, *cuadrantes impracticables* y *cuadrantes practicables*, respectivamente. Como es obvio, nunca nos interesará explorar una dirección dentro de un cuadrante impracticable. El objetivo de mantener información sobre los cuadrantes es reducir la dimensionalidad de la búsqueda de nuevos LSRs y por tanto acortar la duración del proceso de construcción automática del mapa cognitivo. En el capítulo siguiente veremos que también es posible definir mecanismos de planificación utilizando el concepto de cuadrante.

Un cuadrante se premiará siempre que se haya cubierto con éxito un trayecto hasta o desde el LSR al que pertenece. Un cuadrante se castigará por dos posibles motivos: (a) al detectar un MLS se castigará el cuadrante correspondiente a la dirección de partida del último LSR visitado; (b) al crear un LSR se castigarán todos los cuadrantes cuya máxima medida sensorial esté por debajo de cierto umbral *D*; esto evita explorar direcciones que conducen directamente a una pared. La ecuación 4.2 permite obtener el cuadrante de un LSR dado un vector de dirección (se utiliza tanto para premiar como para castigar). La ecuación 4.3 determina todos los cuadrantes cuya *máxima medida sensorial* (la mayor medida de todos los sensores que caen dentro de un cuadrante *q*) está por debajo de cierto umbral *D*.

$$c[\vec{d}] = \text{floor}\left[\frac{Q \cdot \text{atan}(\vec{d})}{2\pi} + \frac{1}{2}\right] \tag{4.2}$$

$$\begin{aligned} & s_{max}[q] = \max_{c[\vec{d}_k]=q}\left\{s_k\right\} \\ & \text{castigar } q \Leftrightarrow s_{max}[q] < D \end{aligned} \tag{4.3}$$

donde *Q* es el número de cuadrantes, s_k es la medida del sensor *k* y $\vec{d}_k$ es la dirección a la que apunta el sensor *k*.

Trabajando con ángulos, la ecuación 4.4 permite la determinación del cuadrante más afín a un ángulo α.

$$\begin{aligned} &\omega_q = \left|\alpha - \beta_q\right| \Leftrightarrow \left|\alpha - \beta_q\right| < \pi \\ &\omega_q = 2\pi - \left|\alpha - \beta_q\right| \Leftrightarrow \left|\alpha - \beta_q\right| > \pi \\ &\text{seleccionar } q \text{ tal que } \omega_q = min\left\{\omega_i\right\} \forall i \end{aligned} \qquad (4.4)$$

siendo β_q el ángulo director (el ángulo de la bisectriz) del cuadrante q.

4.4 Grafo de lugares sensorialmente relevantes (GLSR)

Para incrementar la eficiencia y evitar los MLS es imprescindible disponer de un modelo global del entorno que constituirá el nivel cognitivo del robot. En el resto del capítulo presentaremos la *exploración basada en cuadrantes*, un nuevo método de construcción y mantenimiento de un modelo topológico computacionalmente abordable, robusto y sencillo de manipular. Este modelo se apoya en (a) un nivel reactivo basado en las cargas de potencial artificial ampliado con la capacidad de añadir cargas ficticias, (b) el concepto de gradiente sensorial como detector de LSRs (c) la detección de MLSs y (d) la idea de cuadrante. Es de resaltar que el modelo se está actualizando permanentemente, incluso cuando no se está en modo de exploración, si bien el modo de exploración es más exhaustivo. En este capítulo expondremos el método de exploración basada en cuadrantes junto con algunas ideas para el refinamiento de los modelos construidos. El capítulo siguiente lo dedicaremos al problema de cómo utilizar los modelos construidos para la elaboración de planes.

Para relacionar los diferentes LSRs en un modelo conjunto utilizaremos una estructura en forma de grafo, que denominaremos *grafo de lugares sensorialmente relevantes* (GLSR), cuyos nodos serán LSRs. Cuando un LSR es directamente alcanzable desde otro (es decir, entre ellos no media ningún MLS) los uniremos a través de un arco. El coste de cada arco será el esfuerzo estimado —inicialmente en distancia o en tiempo de navegación— para llegar de un nodo a otro. En definitiva, un arco del GLSR indica que entre los LSRs que conecta no existe ningún MLS. Obsérvese que según esta definición los arcos del GLSR no están exentos de solapar con algún obstáculo; lo que sí nos garantizan es que el navegador utilizado (bien sea el propuesto en esta tesis u otro cualquiera) es capaz por sí mismo de cubrir el trayecto entre los dos nodos.

Cuando la detección basada en el gradiente sensorial se dispara en lugares muy próximos en el espacio (por debajo de cierta distancia umbral) se agrupan en un único nodo. Esta agrupación intenta contener el crecimiento del número de nodos y, aún más importante, incrementa la conectividad del grafo, lo que redunda en la obtención de caminos más eficientes.

Con el fin de establecer enlaces entre nodos, el sistema recuerda siempre el último LSR visitado. Cuando detecta un nuevo LSR, establece un enlace desde el último LSR visitado hasta el nuevo. Del mismo modo podemos establecer el enlace recíproco (obteniéndose un grafo no dirigido), aunque en rigor —como se verá más adelante— no podemos asegurar la reversibilidad del camino en todos los casos. Recordamos que los mínimos locales *son dependientes de la colocación del objetivo*, y por ello suelen aparecer sólo en uno de los dos sentidos de recorrido de un arco. Si utilizamos un grafo no dirigido incrementamos el número de rutas posibles, y por tanto la potencia del modelo, pero a costa de un mayor riesgo de error.

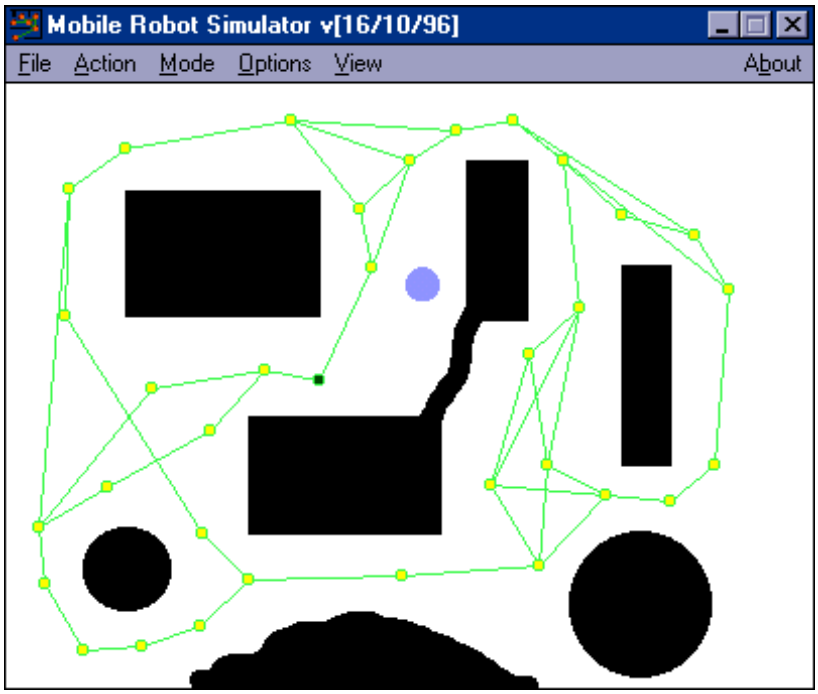


Figura 4.1. Ejemplo de GLSR

Cuando durante una misión el sistema detecte un MLS deberá olvidar el último LSR visitado, ya que, si caemos en un MLS, podemos asegurar que el sistema de navegación no ha podido encontrar una ruta hacia un nuevo LSR con el último rumbo fijado. Cuando se

encuentra un nuevo LSR y no hay último LSR visitado, simplemente se asigna el nuevo LSR como último visitado y no se crea arco. Este procedimiento genera grafos que pueden ser no conexos, pero garantiza que siempre que un arco enlace dos LSRs, el sistema de navegación reactivo podrá cubrir el trayecto, salvo que haya habido cambios en el entorno.

En la figura 4.1 puede verse el aspecto que presenta un GLSR parcialmente construido.

4.5 Evolución temporal del GLSR

Un aspecto importante del GLSR es su evolución temporal. Dado que trabajamos en entornos cambiantes, es preciso que el grafo sea capaz de evolucionar con el paso del tiempo. Es posible que determinados arcos dejen de existir o que nuevos arcos aparezcan debido a que algunos objetos (mesas, sillas, etc.) cambien de lugar. Nos interesa encontrar un mecanismo que ayude a estimar la calidad de un arco, o expresado formalmente, la probabilidad de que dicho arco se encuentre ocupado en una misión posterior. Esta probabilidad podrá añadirse al arco como costo adicional [Kumpel & Serradilla, 1990], o podrá fijarse un umbral de probabilidad por encima del cual el arco no será considerado en el proceso de planificación. Trataremos el tema en profundidad en el capítulo siguiente.

4.6 Identificación de nodos

El gradiente sensorial nos permite detectar LSRs. Pero la detección de LSRs no basta; necesitamos también *identificarlos*. El gradiente sensorial es como una alarma que avisa de que el robot está en un lugar interesante, pero es preciso averiguar qué lugar es ese. Para identificar un LSR disponemos de dos tipos de información: sus coordenadas aproximadas y la imagen sensorial promedio almacenada.

Algunos investigadores han utilizado recientemente métodos de reconocimiento de formas para la identificación de lugares de interés. [Mataric, 1992], [Kurz, 1993; 1996] y [Walker *et al*, 1993] proponen el uso de mapas autoorganizados para este propósito; Kurz también utiliza técnicas clásicas, como la elección del vecino más cercano. En nuestro trabajo hemos utilizado este último método, ya que resuelve holgadamente el problema.

En síntesis, nuestro algoritmo de identificación de nodos se basa en calcular el LSR que haga mínima la expresión 4.5 con respecto a la posición actual *p*.

$$d_e(l,p) = \sqrt{\Gamma\left((l_x - p_x)^2 + (l_y - p_y)^2\right) + \sum_{i=0}^{N-1}\left(s_i(l) - s_i(p)\right)^2} \quad (4.5)$$

donde *l* es el LSR a considerar, *p* es la posición actual del robot, $s_i(l)$ se refiere a la medida promedio del sensor *i* desde el LSR, $s_i(p)$ se refiere a las medidas sensoriales desde la posición actual, *N* es el número de sensores y Γ es un número real que define la importancia relativa asignada a la posición estimada del robot frente a la información sensorial. Si tomamos Γ=1, esta expresión no es otra cosa que la distancia euclídea entre los vectores de características de los nodos, estando compuesto el vector de características por las medidas de los sensores junto con las coordenadas de posición.

Un sencillo algoritmo calculará, para todo *l* del GLSR, la medida $d_e(l,p)$, y seleccionará el nodo *l* que proporcione el menor valor. Si $d_e(l,p)$ es mayor que cierto valor umbral, concluimos que el LSR detectado no se encuentra en el modelo, en cuyo caso deberemos crearlo.

Siempre que un nodo sea identificado y no sea igual al último nodo visitado, se actualizará su información para contemplar la información obtenida en la posición actual, de modo que

- Se añadirá como nuevo arco el enlace entre el último LSR detectado y el actual.
- Se promedia cada componente del mapa sensorial con la información sensorial actual para obtener un *mapa sensorial promedio*.

4.7 Construcción automática del GLSR

Aunque el GLSR se está construyendo en todo momento, es útil disponer de mecanismos específicos y sistemáticos de construcción que sean capaces de generar en pocos pasos un modelo previo del entorno y que aprovechen los momentos de inactividad del sistema para refinar este modelo. Hablamos entonces de que el sistema es capaz de realizar tareas de aprendizaje autónomo, problema del que se han ocupado numerosos autores [Shen, 1994; Weiss & Kulikowski, 1991; Brooks & Mataric, 1993; Connell & Mahadevan, 1993; Maes &

Brooks, 1990; Mataric, 1992; Kurz, 1996] y que constituye uno de los grandes retos actuales de la IA.

Para afrontar este problema hemos desarrollado un conjunto de algoritmos [Serradilla & Maravall, 1997] que utilizan información de diferentes niveles según exponemos en los apartados siguientes.

4.7.1 Exploración basada en cuadrantes

Como ya anticipamos, el algoritmo de exploración basada en cuadrantes construye exhaustivamente un modelo topológico del entorno. Para ello utiliza la información de los cuadrantes, la información sensorial, la detección de LSRs y MLSs y el método de navegación implementado en el sistema. En nuestro caso el método de navegación es el del potencial artificial en modo rumbo, es decir, en su versión más robusta. Téngase presente que cuando en la descripción del algoritmo hablamos de "avanzar" nos referimos a moverse utilizando el navegador con el rumbo fijado, de modo que el propio navegador se encarga automáticamente de eludir cualquier obstáculo fijo o móvil que pudiera existir en la dirección de movimiento.

La estrategia de exploración es la siguiente:

EXPLORE ()

1) Hacer LastLSR=NULL. Elegir al azar una dirección de exploración (rumbo).
2) Si se detecta MLS,
   a) Si LastLSR==NULL ir a 1, de lo contrario
   b) Marcar como impracticable el cuadrante del último LSR visitado (LastLSR) correspondiente al rumbo actual.
   c) Si LastLSR ha sido visitado cierto número máximo de veces, ir a 1, de lo contrario
   d) Modificar rumbo para apuntar a LastLSR (es decir, para regresar al último LSR visitado); Hacer LastLSR=NULL.
3) Si se detecta LSR,
   a) Identificar el LSR detectado (utilizando el procedimiento basado en la ecuación 4.5). Si el LSR identificado es igual a LastLSR, salir del punto 3, en caso contrario
   b) Si el LSR no existe en el GLSR (la identificación ha fracasado),

   i) Crear LSR en GLSR; marcar como impracticables los cuadrantes del LSR con máxima medida sensorial menor que un cierto umbral *D*.
   ii) Añadir el nuevo LSR al GLSR. Si LastLSR≠NULL crear un arco con LastLSR.
c) Si LastLSR≠NULL, marcar como practicable el cuadrante de LastLSR correspondiente a la dirección de exploración actual. Alternativamente también pueden marcarse como practicables los cuadrantes correspondientes a la dirección que une LastLSR y LSR (que no tiene por qué coincidir exactamente con la de exploración).
d) Elegir de entre los cuadrantes no explorados del LSR el más afín a la dirección actual del robot utilizando las ecuaciones 4.2 ó 4.4.
e) Elegir al azar una dirección de exploración dentro de ese cuadrante. Si no hay ningún cuadrante no explorado la elección se efectuará entre todos los cuadrantes practicables. En el caso (extremadamente improbable) de que todos los cuadrantes sean impracticables, elegir al azar.
f) Hacer LastLSR=LSR.

4) Avanzar un paso en la dirección de exploración e ir a 2.

El motivo de que sólo se ejecuten las acciones del punto 3 cuando LastLSR≠LSR es que la detección basada en el gradiente suele generar ráfagas porque en una zona de alto gradiente el detector se activará varias veces. Esta comprobación nos evita cálculos innecesarios.

Una versión alternativa de este algoritmo incluye la idea de refuerzo de cuadrantes. En lugar de simplemente marcar los cuadrantes con éxito como hábiles, éstos pueden premiarse —incrementando cierto *valor de refuerzo*— en el punto 3.c, es decir, cada vez que se obtiene un éxito utilizando dicho cuadrante. Desde esta perspectiva, nos interesará, siempre que alcancemos un LSR ya existente, elegir como nuevo rumbo —en el punto 3.d— el *cuadrante practicable con menos refuerzo*. Esto obligará al sistema a elegir direcciones de exploración poco conocidas, haciendo aún más exhaustivo el procedimiento de exploración. Esta versión del algoritmo, sin embargo, presenta el inconveniente de provocar un gran número de retornos del robot por una dirección parecida a la de llegada al nodo, lo que da un cierto aire de ineficiencia al sistema. Una mejora podría consistir en combinar ambos procedimientos, estableciendo un compromiso entre el cuadrante más afín a la dirección actual del robot y el menos explorado, siempre de entre los hábiles.

En las figuras 4.2 y 4.3 podemos ver un modelo construido utilizando este procedimiento. En la figura 4.2 se muestran los LSRs con sus cuadrantes y en la figura 4.3 la interconexión de LSRs. Obsérvese que un arco del GLSR *puede interceptar un obstáculo*; esto no supone un problema y significa que el sistema de navegación es capaz por sí solo de evitar este obstáculo sin quedar atrapado en un MLS.

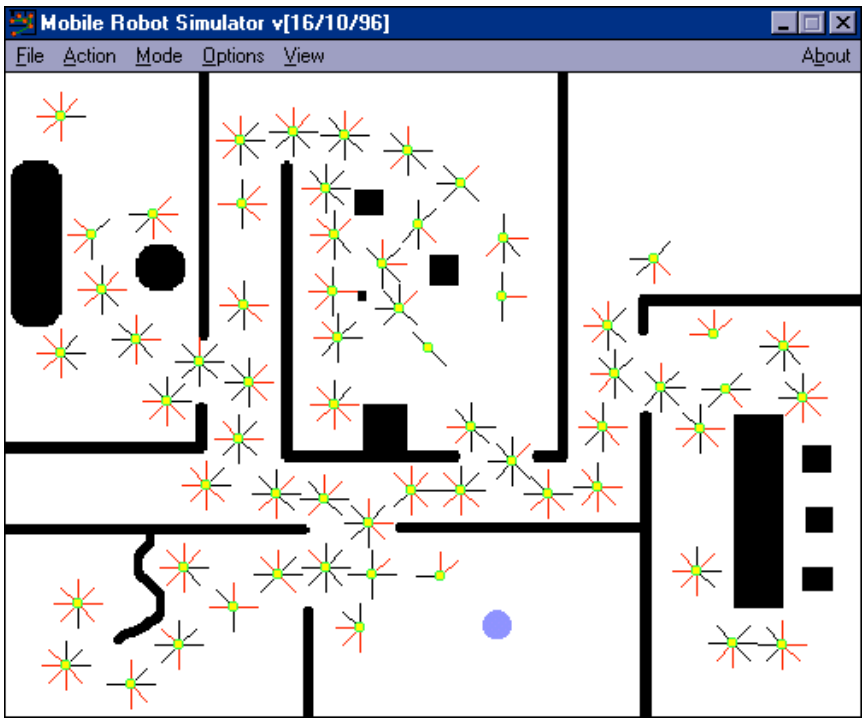


Figura 4.2. LSRs y cuadrantes generados en una misión de exploración. Los cuadrantes impracticables se muestran en rojo; los cuadrantes practicables, en negro. Las posiciones en blanco corresponden a cuadrantes sin explorar.

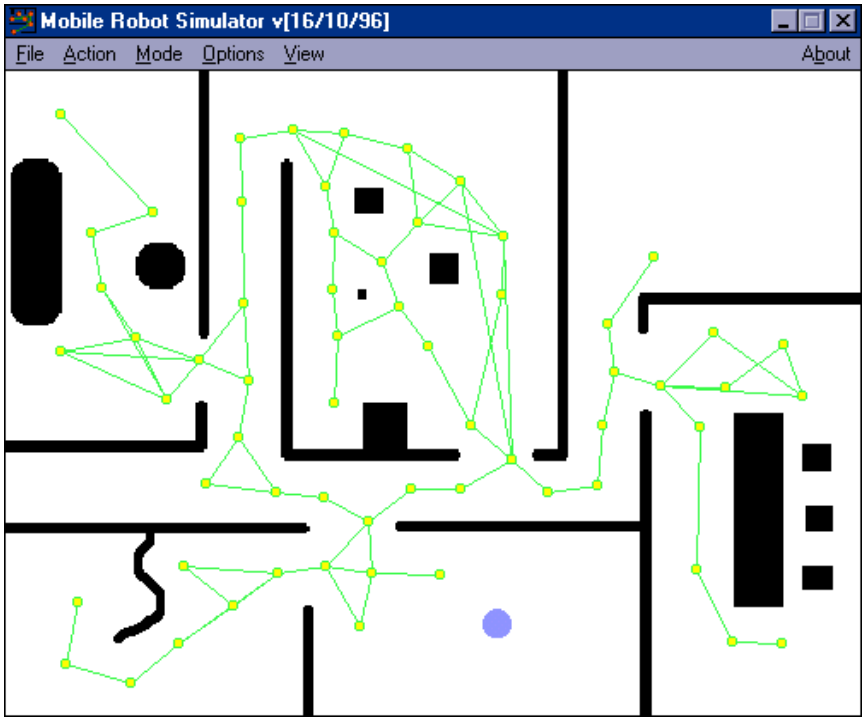


Figura 4.3. GLSR generado en la misma misión de exploración

4.7.2 Refinamiento del GLSR utilizando los mapas sensoriales de sus nodos

Un sistema para mejorar el modelo construido mediante el algoritmo de exploración basada en cuadrantes es utilizar la idea de visibilidad (en la sección 5.6.1 definiremos formalmente este concepto) entre dos posiciones del mundo para conectar nodos que sean visibles entre sí.

El método de refinamiento es muy simple: comprobar, para todos los pares (A,B) de nodos del GLSR (N^2 comprobaciones), si la medida de la imagen sensorial promedio de A que apunta en la dirección de B es menor que la distancia entre los nodos A y B. Si esto sucede, B es visible desde A, y podremos añadir al GLSR un arco que una A con B.

En la figura 4.4 puede verse el resultado de aplicar este procedimiento al modelo de la figura 4.3. En rojo se muestran los nuevos arcos añadidos.

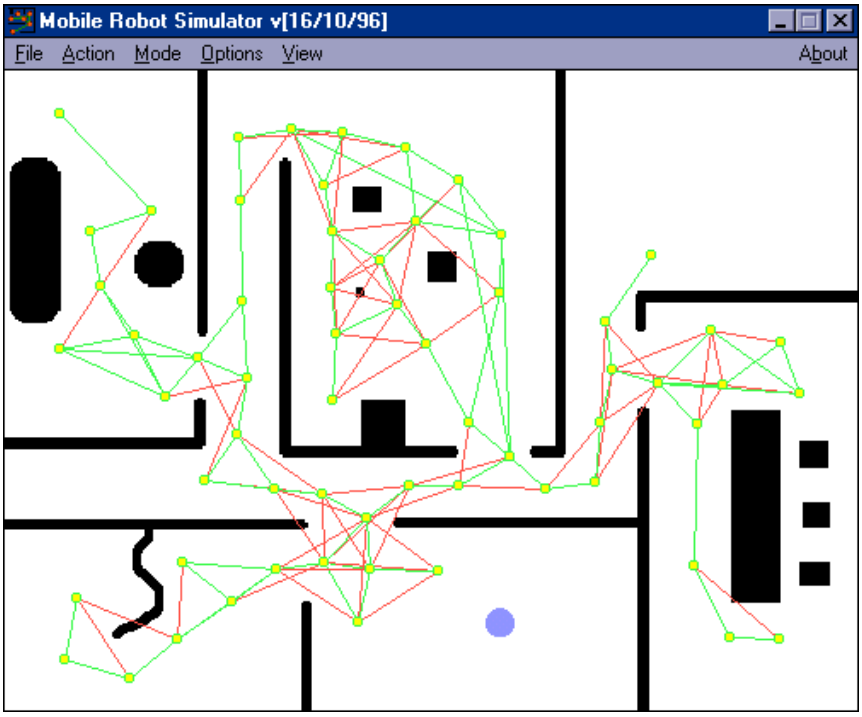


Figura 4.4. GLSR completado con el procedimiento de refinado

4.8 Reflexiones sobre direccionalidad del grafo y cuadrantes relativos

Nuestro sistema perceptivo es dependiente de la orientación. Muchas veces hemos experimentado extrañeza al llegar, desde una perspectiva nueva, a un lugar conocido; no es una tarea sencilla recorrer una ruta frecuentada en sentido inverso al habitual. Estos fenómenos se deben a que el reconocimiento de un lugar depende del punto de vista del individuo.

De la misma manera, la detección de LSRs es, en principio, dependiente de la dirección de llegada del robot. Sin embargo, el método de detección utilizado —el módulo del gradiente sensorial— unido a la simetría sensorial del robot, hacen posible el reconocimiento de un LSR desde casi cualquier ángulo de aproximación. Hemos explotado esta circunstancia, aún a sabiendas de que la hipótesis de grafo no-dirigido no es cierta en todos los casos, porque enriquece considerablemente el modelo. Esto supone que en algunas (raras) ocasiones un LSR no será detectado. Afortunadamente, los mecanismos de planificación que propondremos en el capítulo 5 son suficientemente robustos para que esto no sea problema.

Como se vio en el capítulo 2, el propio sistema de navegación basado en el potencial artificial es no-reversible. Un mínimo local que aparece en una dirección puede no aparecer (y de hecho normalmente no aparecerá) en la dirección contraria (ver figura 4.4). Esto puede ocasionar en alguna (también rara) ocasión que el sistema construya un arco del GLSR que no pueda recorrerse en la dirección contraria. De nuevo la pequeña probabilidad de este hecho compensa con el incremento de potencia de planificación que proporciona un grafo no-dirigido, y de nuevo los planificadores que utilizaremos son suficientemente potentes como para dar cuenta de este problema. Una hipótesis menos restrictiva podría asignar una probabilidad de ocupación mayor a la dirección de retorno, ya que de hecho el sistema no ha verificado la validez del arco en esta dirección.

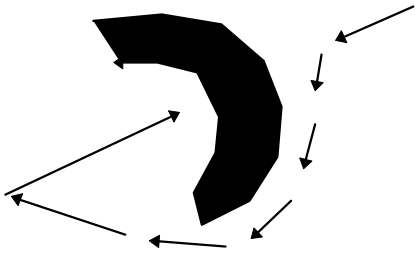


Figura 4.4. Una situación no reversible para el método del potencial

4.9 Resultados

En el capítulo 4 hemos tratado de la concepción y construcción de un modelo topológico del entorno en el que se mueve el robot. A lo largo de la exposición de los métodos hemos mostrado algunos resultados de su aplicación, obtenidos a partir del simulador desarrollado para esta tesis. Nuestras aportaciones originales en torno a la creación de modelos son:

a) Hemos introducido la noción de cuadrante de exploración. Este concepto ha mostrado ser una potente herramienta para la construcción automática de modelos. En el capítulo siguiente veremos que también es útil para definir estrategias de planificación reactivas.

b) Hemos definido el concepto de GLSR, un modelo topológico que relaciona entre sí los distintos LSR detectados utilizando las técnicas del capítulo 3.

c) Hemos tratado el problema de la identificación de nodos con información odométrica imprecisa, utilizando un clasificador basado en la distancia euclídea, que tiene en cuenta la posición aproximada del robot y la información sensorial captada, por un lado, y la posición y la imagen sensorial promedio almacenadas en el LSR, por otro.

El clasificador permite determinar, al detectar un LSR utilizando el gradiente sensorial, si el LSR existe en el modelo previo y, en ese caso, de qué LSR se trata.

d) Hemos definido procedimientos de construcción del GLSR en el transcurso de las misiones del robot. Para ello son importantes los conceptos de *último LSR visitado* y de *mínimo local severo*.

e) Hemos desarrollado —apoyándonos en la idea de cuadrantes de exploración— un algoritmo de exploración exhaustiva del entorno que permite la construcción incremental del GLSR.

f) Hemos desarrollado un mecanismo de refinamiento del GLSR utilizando la idea de visibilidad entre nodos.

g) Finalmente, hemos realizado una serie de reflexiones sobre direccionalidad en los modelos de robótica móvil, argumentando la conveniencia de un modelo bi-direccional siempre que el navegador y los mecanismos de planificación sean suficientemente robustos.

5. Nivel cognitivo. Elaboración, ejecución de planes y mantenimiento del modelo.

5.1 Introducción

En este capítulo propondremos tres métodos de planificación. El primero de ellos utiliza información local almacenada en los LSRs y podría catalogarse como una planificación de tipo reactivo; es la *planificación basada en cuadrantes*. El segundo utiliza un modelo de información global —el GLSR— para obtener un plan de ruta completo para la misión propuesta mediante un algoritmo A*. Este plan de ruta es un plan tentativo que incorpora la posibilidad de replanificar si algo falla; es lo que se conoce como *planificación incremental o interfoliar*. Finalmente, el tercer método propuesto —la *búsqueda en oleadas*— se basa en la idea de *plan internalizado* [Payton, 1991], y la implementa adjudicando a cada LSR una dirección preferente de movimiento que garantice estar en el camino óptimo hacia la meta.

Mientras que la planificación utilizando el método A* ha sido extensamente utilizada en los enfoques clásicos [Lozano-Pérez & Wesley, 1979; Mitchell, 1988; Elfes, 1989; Kumpel & Serradilla, 1989; Storer & Reif, 1994], los otros dos métodos —planificación basada en cuadrantes y planificación en oleadas— son aportaciones de esta tesis, y obedecen a la propuesta de Agre y Chapman [Agre & Chapman, 1990] de *planes expresados mediante sugerencias*. En un plan expresado mediante sugerencias las tareas a realizar no se manifiestan como un conjunto de instrucciones a ejecutar mecánicamente, sino como un conjunto de indicaciones o *sugerencias* a un agente, que debe ser suficientemente autónomo como para

sacar partido de estas indicaciones, sugerencias o consejos. Desde este punto de vista, un plan es un sistema de apoyo a la decisión, y no la decisión en sí misma, la cual dependerá también de otros factores locales y de las características del agente que interpreta el plan.

5.2 Sugerencias locales y sugerencias globales

Hemos introducido una distinción entre dos tipos de sugerencias, que hemos denominado *locales* y *globales*. Una *sugerencia local* es una indicación de ruta que es localmente óptima, es decir, intenta maximizar la probabilidad de que el robot encuentre la meta ateniéndose a la información local del sistema (los cuadrantes del último LSR detectado, la dirección actual de movimiento, etc). Una *sugerencia global* utiliza un modelo global del entorno —en nuestro caso el GLSR— para establecer la indicación *óptima* en cada estado. Si el robot sigue con éxito todas las sugerencias globales que encuentre, entonces podemos garantizar que llegará a la meta y además lo hará siguiendo la ruta óptima.

La ventaja de la planificación utilizando sugerencias globales es que permite desarrollar el camino óptimo. La ventaja de la planificación utilizando sugerencias locales es que, si bien no garantiza encontrar la mejor solución, funciona aún en ausencia total de modelo. Esto la hace especialmente recomendable en etapas tempranas de funcionamiento en las que aún no se ha desarrollado completamente el GLSR.

Cuando intentamos encontrar un plan global y se comprueba que el modelo global no es suficiente para la misión encomendada (no existe ningún nodo del GLSR próximo al punto de destino), el sistema cambia automáticamente a una planificación basada en sugerencias locales. Otra ventaja de este modo es que durante la misión se continúa actualizando el modelo global, de manera que en misiones posteriores hacia la misma zona podrá utilizarse la planificación basada en sugerencias globales.

5.3 Planificación basada en cuadrantes

En el capítulo 4 se expuso la noción de cuadrante y se comentó cómo se actualizaban sus valores. Recordamos aquí que un cuadrante podía estar en uno de tres estados: *inexplorado*, *practicable* o *impracticable*. Para la planificación basada en cuadrantes inicializaremos los

cuadrantes practicables e inexplorados como *no usados*, y los cuadrantes impracticables como *usados*. Cada vez que el robot detecte un LSR, utilizaremos el estado de uso de sus cuadrantes, la dirección actual de movimiento y la dirección en la que se encuentra la meta para determinar el rumbo que tomará el robot. Una vez que un cuadrante es seleccionado se marca como *usado*. Esta marca se mantendrá hasta que la misión finalice o sea abortada, y significa que dicho cuadrante no volverá a ser elegido durante la misión a menos que todos los cuadrantes del LSR hayan sido usados.

La idea básica de la planificación basada en cuadrantes consiste en elegir, cada vez que se detecta un LSR, el cuadrante no usado cuya orientación es más parecida a la dirección de la meta. Tras esto asignaremos al rumbo del robot la orientación del cuadrante elegido. Cuando se elige un cuadrante se marca como usado, de modo que la siguiente vez que el robot llegue a ese LSR no pueda volver a elegirlo (lo que sin duda produciría un bucle infinito). Cuando el sistema detecte un MLS, se fijará el rumbo hacia el último LSR visitado con objeto de continuar la exploración desde él.

Esta idea simple, sin embargo, produce demasiados reintentos en la dirección de la meta, producidos tanto por LSR cercanos como por cuadrantes del mismo LSR que apuntan en direcciones parecidas. Esto hace que el sistema se comporte de modo ineficiente cuando la meta se encuentra tras un MLS (figura 5.1).

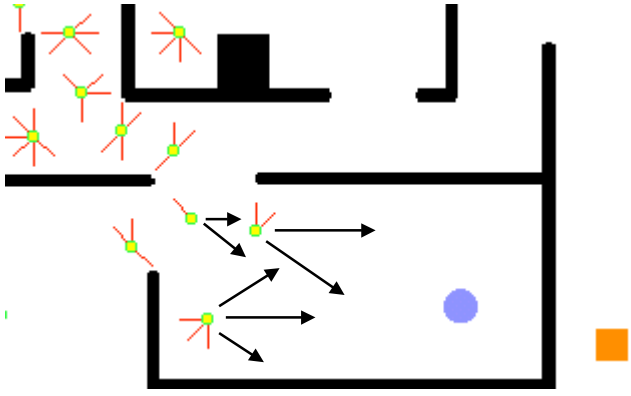


Figura 5.1. Situación que produciría demasiados reintentos en caso de utilizarse el enfoque simple

Para mejorar el planteamiento inicial hemos utilizado una idea tomada del campo de las redes de neuronas, la *adición de momento*, utilizada en el entrenamiento de perceptrones multicapa con descenso del gradiente [Hertz *et al*, 1991; Hinton, 1990]. En nuestro caso, el

momento viene dado por la dirección actual de movimiento del robot. Para elegir la dirección recomendada de movimiento utilizaremos la ecuación 5.1.

$$\vec{r} = \vec{g} + \delta \cdot \vec{v} \qquad (5.1)$$

donde $\vec{g}$ es un vector unitario en la dirección de la meta, $\vec{v}$ es un vector unitario en la dirección actual de movimiento del robot (el momento) y δ es un factor que permite ajustar la tendencia conservadora a continuar en la dirección actual. En nuestro trabajo hemos utilizado δ=0,9.

Una vez determinada la dirección deseada de movimiento, el algoritmo escoge, de entre los cuadrantes no usados, el cuadrante cuya orientación es más parecida a la dirección deseada según la ecuación 5.2.

$$\begin{aligned} &abs(\alpha_d - \alpha_i) \Leftrightarrow abs(\alpha_d - \alpha_i) \leq \pi \\ &2\pi - abs(\alpha_d - \alpha_i) \Leftrightarrow abs(\alpha_d - \alpha_i) > \pi \end{aligned} \qquad (5.2)$$

siendo α_d el ángulo de la dirección deseada, es decir, $atan(\vec{r})$, y α_i el ángulo director del cuadrante a considerar.

Si un LSR ha utilizado todas sus posibles sugerencias (tiene usados todos sus cuadrantes) con objeto de alcanzar una meta particular y ninguno ha sido útil, el nodo pierde su capacidad informativa para el resto de la misión; a partir de ese momento no se tendrá en cuenta a efectos de planificación. Cuando el robot detecta uno de estos LSR simplemente continúa con la dirección que llevaba.

Un último detalle es que llevaremos la cuenta de las veces en que un LSR ha sido visitado. Superado cierto número de visitas, no se fijará el rumbo hacia ese LSR al encontrar un MLS tras haberlo visitado. En lugar de esto, el nuevo rumbo se elegirá al azar.

Formalmente, al algoritmo de planificación basada en cuadrantes es el siguiente:

QUAD-PLAN (GOAL)

1) Marcar como usados los cuadrantes impracticables y como no usados el resto. Fijar rumbo en dirección a GOAL. Hacer LastLSR=NULL.

2) Hasta que GOAL sea visible (ver apartado 5.6.2),
   a) Si se detecta LSR y éste dispone de algún cuadrante no usado
      i) Calcular dirección deseada utilizando la ecuación 5.1.
      ii) Determinar, de entre los cuadrantes no usados, aquél cuyo ángulo director sea más parecido a la dirección deseada, utilizando la ecuación 5.2; fijar rumbo en la dirección del cuadrante y marcarlo como usado.
   b) Si se detecta MLS
      i) Si hay último cuadrante visitado (LastLSR≠NULL) y éste no ha superado el máximo número de visitas, fijar rumbo hacia LastLSR.
      ii) De lo contrario, elegir nuevo rumbo al azar
   c) Avanzar un paso en la dirección indicada por el rumbo.

En la figura 5.1 se muestra el resultado de una misión de planificación basada en cuadrantes. Antes de la misión no existía modelo del entorno. En la figura 5.2 se muestra el modelo construido en el transcurso de la misión.

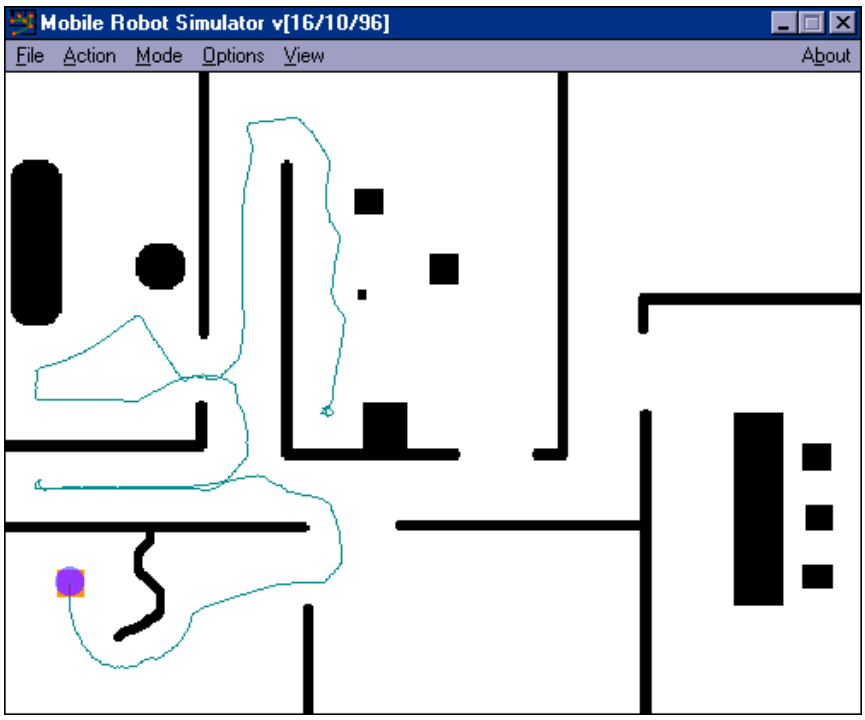


Figura 5.1. Resultado de una planificación basada en cuadrantes

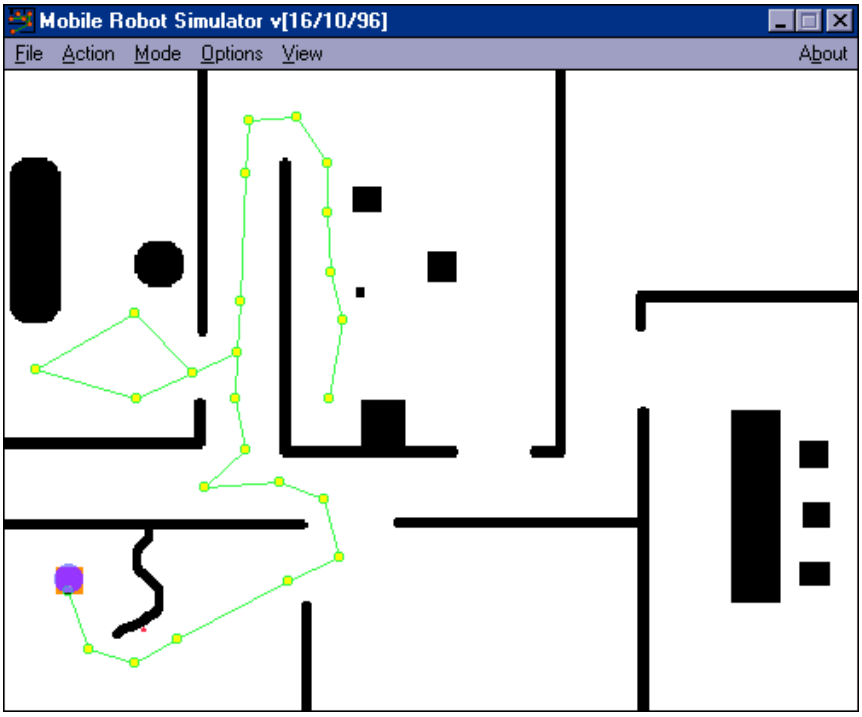


Figura 5.2. Modelo construido tras la misión de la figura 5.1.

5.4 Planificación utilizando el GLSR

Una vez que disponemos de un modelo en forma de GLSR que refleja la estructura del entorno del robot, podemos planificar nuevas misiones aprovechando este conocimiento. Dado que el mundo es cambiante, los planes que podamos obtener serán planes tentativos, y habrá que confrontarlos con el mundo real para asegurarnos de que pueden llevarse a cabo. Por ello, en este nivel de abstracción —el de planificación global sobre el GLSR— debemos efectuar dos tareas:

1. obtener un plan
2. ensayar el camino tentativo obtenido

Si el plan fracasa, es decir, si el camino tentativo no puede completarse, será debido a que alguna de las subtareas que lo componen no se pudo completar; en este caso, será necesario actualizar el modelo del mundo para reflejar los cambios en la zona donde la subtarea fracasó, y obtener, aplicando recursivamente el mismo algoritmo, un plan alternativo. Esto es lo que se conoce como *planificación incremental o interfoliar (interleaved planning).*

Para la obtención del plan se han utilizado dos métodos diferentes. El primero de ellos, basado en el algoritmo A*, resume una tradición clásica de planificación de tareas de razonamiento espacial, referida por Agre y Chapman como *plan como programa (plan-as-program)*. El segundo, situado en una perspectiva más moderna (*planificación expresada como sugerencias*), se basa en el cálculo para cada nodo del sucesor óptimo, es decir, el sucesor que tendría dicho nodo si calculásemos el camino óptimo entre él y el objetivo. Para ello hemos desarrollado un algoritmo que permite efectuar este cálculo con un reducido coste computacional.

*5.4.1 Mediante A**

Para la obtención de un plan podemos utilizar un algoritmo A* sobre el GLSR. Este algoritmo clásico [Nilsson, 1987] encuentra el camino óptimo que conecta dos nodos en un grafo. El camino obtenido consta de una secuencia de nodos (LSRs) que deberá ir alcanzando secuencialmente el robot. El encargado de conducir al robot entre cada par de nodos será el sistema de navegación local. El proceso de construcción del GLSR nos garantiza —salvo grandes cambios en el entorno— que entre cada par de nodos no existirá ningún mínimo local en que el sistema pueda quedar atrapado.

El ejecutivo del plan obtenido con A* sería el siguiente:

1) Obtener el plan.
2) Mientras quede algún nodo del plan por recorrer,
   a) Fijar objetivo para el navegador.
   b) Utilizar el navegador en modo meta para alcanzar el objetivo. Si se detecta MLS,
      i) Castigar e inhabilitar arco correspondiente.
      ii) Ir a 1.
3) Utilizar el navegador en modo meta para atracar en la posición objetivo.

En la figura 5.3 puede verse el resultado de una misión con A* sobre el grafo de la figura 4.3 del capítulo anterior.

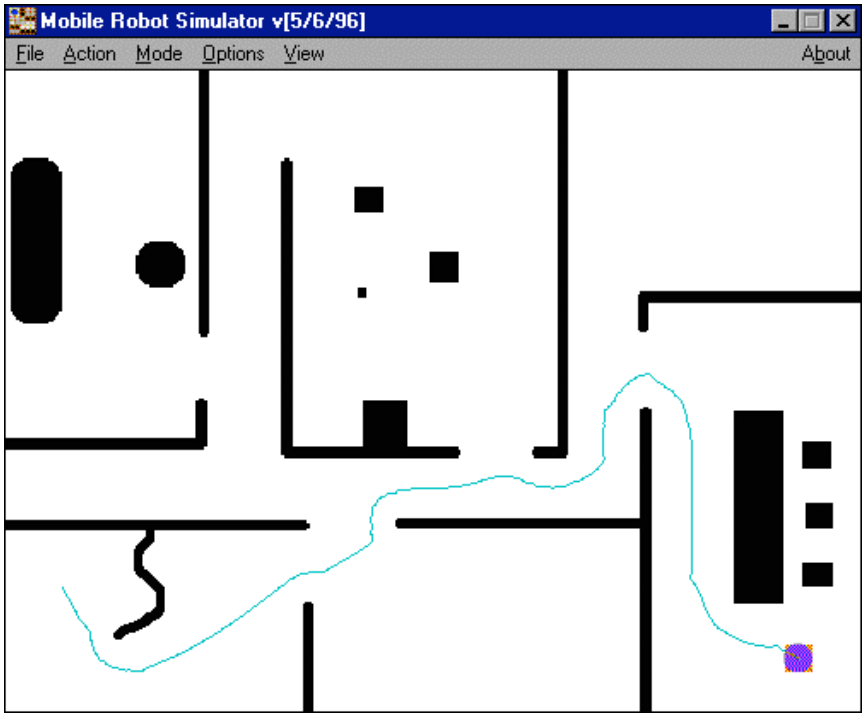


Figura 5.3. Misión realizada utilizando A sobre el grafo de la figura 4.3*

5.4.2 Mediante exploración en oleadas

Como ya anticipábamos, la exploración en oleadas se inspira en la propuesta de *planes expresados mediante sugerencias* de Agre y Chapman, en los *planes universales* de Schoppers [Schoppers, 1987] y en los trabajos de Payton [Payton *et al*, 1990; Payton, 1991] sobre *planes internalizados* y *campos de gradiente*. El objetivo es calcular, para cada nodo del espacio de estados, cuál es la transición óptima (en el sentido de que apunta al siguiente nodo del camino óptimo hacia la meta) desde el nodo considerado. Una vez realizado este proceso tenemos, para cada nodo, una sugerencia sobre la dirección recomendada hacia la que debemos movernos. El conjunto de todas las sugerencias constituye una especie de campo (el campo de gradiente) que "empuja" al robot hacia la solución óptima. La gran ventaja de este enfoque frente al cálculo tradicional de un camino que luego debe ser recorrido es que, si el sistema se pierde —bien por no detectar un nodo, bien por modificaciones en el entorno—, el plan podrá retomarse desde cualquier otro nodo del espacio del problema.

Este planteamiento encaja como anillo al dedo con nuestra propuesta de Lugares Sensorialmente Relevantes, ya que podemos, siempre que se detecte un LSR, (a) identificar de qué LSR se trata, (b) obtener la dirección aconsejada de movimiento almacenada en el LSR

tras el establecimiento del plan internalizado y (c) fijar el rumbo del robot para apuntar hacia dicha dirección.

Para establecer el plan, Payton propone la utilización de un algoritmo A*. Este algoritmo, puesto que está pensado para encontrar el mejor camino entre dos nodos, habría que aplicarlo a cada par de nodos del espacio, es decir N(N-1)/2 veces, lo que supone un consumo de recursos excesivo.

Adaptado a las características del problema, proponemos en esta tesis un nuevo algoritmo que hemos denominado "búsqueda en oleadas" porque puede imaginarse como un conjunto de olas que parten de la meta y recorren el espacio del problema actualizando los valores del mejor nodo sucesor para cada nodo. La primera ola parte de la meta y se propaga a todos los nodos que tienen a ésta como sucesora. Estos nodos se orientan entonces hacia la meta (actualizando un campo del nodo que indica cuál es el sucesor recomendado en cada momento) y se acumula en ellos el coste del camino. Este proceso se repite ahora para todos los nodos que están en la ola, con lo que la ola se expande un paso más, y así hasta que todos los nodos del espacio hayan sido recorridos. En realidad, cada nuevo nodo genera una nueva ola que morirá al ir llegando a nodos que ya han sido visitados.

Si la función de coste utilizada no cumple la restricción de la monotonía [Nilsson, 1980], el espacio de búsqueda presentará repliegues que harán posible que una ola llegue a determinado nodo antes que otra que partió de un nodo más cercano (en lo que se refiere a número de nodos entre ellos). Debido a estos repliegues es preciso observar, siempre que una ola llegue a un nodo ya visitado, si el coste acumulado de la ola es menor que el que ya tenía el nodo; de ser así, se modifica el mejor sucesor del nodo y se actualiza su coste acumulado. Además, puesto que los valores que produjese la propagación de la ola anterior ya no son válidos, se propagará una nueva ola desde el nodo modificado. Si los nodos generadores de olas y la función de coste son seleccionados adecuadamente la probabilidad de aparición de repliegues es mínima, y la eficiencia del algoritmo es muy elevada.

Naturalmente este algoritmo sólo es útil para aquellos espacios de problema en los que el grafo se conozca completamente al realizar la búsqueda, aunque limitando el alcance máximo de una ola se podría extender a cualquier tipo de espacio. Sin embargo, en el problema que

nos ocupa —la planificación en el GLSR— el espacio es completamente conocido en el momento de la búsqueda y el algoritmo es en este caso directamente aplicable.

Para la implementación del algoritmo utilizaremos los siguientes elementos: una lista de olas donde almacenaremos todos los nodos abiertos en un instante dado, es decir, todas las olas que aún tenemos que propagar, y para cada nodo del espacio de búsqueda (a) un campo con la dirección del mejor sucesor hasta el momento (inicialmente a NULL) y (b) un campo con el coste acumulado del mejor camino hasta el momento (inicialmente a cero).

SWELL-SEARCH (GOAL)

1) Hacer GOOD=NULL y GOODCOST=0 en todos los nodos del grafo.
2) Inicializar WAVES con el nodo GOAL.
3) Mientras WAVES no esté vacía:
   a) Seleccionar el elemento con menor valor en el campo GOODCOST y borrar de la lista WAVES[9].
   b) Para cada uno de los padres del nodo SELECCIONADO (todos aquellos que tienen transición hacia él),
      i) Calcular el costo acumulado desde el nuevo nodo hasta GOAL, utilizando la expresión NEWCOST = COSTE($PADRE_i$, SELECCIONADO) + SELECCIONADO.GOODCOST.
      ii) Si $PADRE_i$ no es GOAL y, o bien no ha sido visitado ($PADRE_i$.GOOD == NULL), o bien habiéndolo sido el nuevo coste calculado es menor que el coste actual (NEWCOST < $PADRE_i$.GOODCOST),
         - Hacer $PADRE_i$.GOOD = SELECCIONADO.
         - Hacer $PADRE_i$.GOODCOST = NEWCOST.
         - Añadir $PADRE_i$ a la lista WAVES por detrás (dado que se ha encontrado un valor mejor para el coste hasta $PADRE_i$, este nuevo valor debe propagarse).

Este algoritmo puede modificarse para tener en cuenta algún criterio heurístico, simplemente sustituyendo, en el punto 3.b.i, la función COSTE por una función COSTE+HEURÍSTICO, de modo similar a como se hace en el algoritmo A*.

[9] Opcionalmente puede elegirse simplemente el primer elemento de la lista; el procedimiento se simplifica y podemos garantizar que el algoritmo seguirá funcionado correctamente, aunque el coste computacional se incrementa. El motivo es que los frentes de onda se propagarán de manera desordenada, aumentando la probabilidad de que se generen nuevos frentes de onda debido a la actualización del coste de un nodo. En grafos pequeños, no obstante, la diferencia de coste computacional no es apreciable.

El ejecutivo del plan obtenido con la búsqueda en oleadas sería el siguiente:

1) Obtener el plan.
2) Fijar rumbo en dirección al LSR accesible más cercano. Si no existe ningún LSR accesible, fijar rumbo en dirección a la meta.
3) Mientras la meta no sea visible,
   a) Si se detecta LSR, actualizar rumbo con la sugerencia ofrecida por el LSR.
   b) Si se detecta MLS,
      i) Castigar e inhabilitar el arco correspondiente.
      ii) Ir a 1.
   c) Utilizar el navegador para avanzar en dirección a la meta.
4) Utilizar el navegador para atracar en la meta.

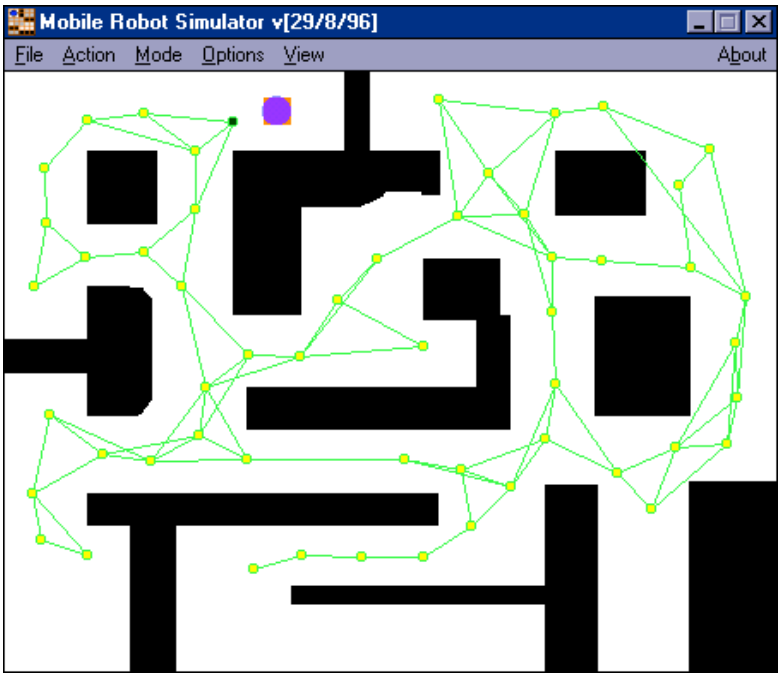


Figura 5.4. Modelo cognitivo previo, construido con los procedimientos de exploración expuestos en el capítulo 4

En la figura 5.5 puede verse un ejemplo de ejecución del algoritmo SWELL-SEARCH sobre el grafo de la figura 5.4. Como puede observarse, cada LSR ha quedado etiquetado con la dirección de su mejor sucesor para alcanzar el nodo objetivo (el nodo que no tiene flecha, junto a la meta). En la figura 5.6 se muestra el camino desarrollado por el navegador del robot utilizando las sugerencias de la figura 5.5.

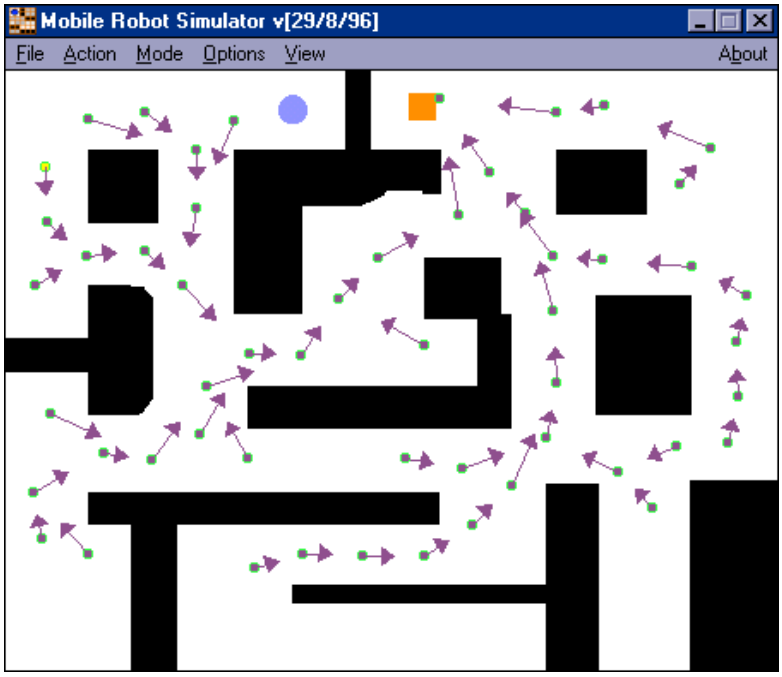


Figura 5.5. Conjunto de sugerencias obtenidas con el algoritmo de búsqueda en oleadas

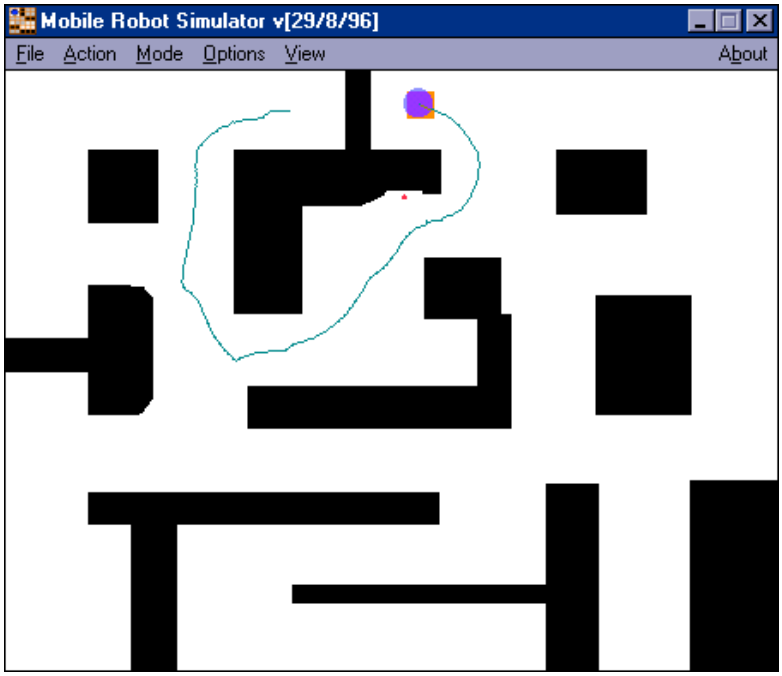


Figura 5.6. Trayectoria obtenida por el navegador siguiendo las sugerencias de la figura 5.5

5.5 Consulta rápida a nodos

En un programa de experimentación, la recuperación de la información de los nodos del GLSR no plantea ningún problema (estamos hablando de grafos de 40 o 50 nodos que pueden

recorrerse linealmente hasta encontrar el que verifica una determinada propiedad), pero, en la vida real, los seres humanos manejamos redes que enlazan cientos de miles de conceptos. Los psicólogos han mantenido tradicionalmente la idea de que estos conceptos se relacionan en una *red semántica*, de tal modo que los items que normalmente deseamos recuperar se encuentran en un entorno de vecindad cercano al del ítem actualmente situado en la memoria a corto plazo. Una exploración muy breve basta para alcanzar el nodo deseado; de lo contrario la búsqueda se abandona.

Al hilo de esta idea, proponemos una "búsqueda semántica" para establecer cuál es el LSR detectado en un instante determinado. Esta búsqueda utiliza un ítem de la *memoria a corto plazo* del robot, el último LSR visitado, para iniciar una exploración en anchura de profundidad limitada (3 o 4 niveles, como máximo, que con un promedio de conectividad en torno a 3 significaría analizar unos 81 nodos como máximo). Si el LSR no se encuentra en esta búsqueda se considerará que se trata de un nuevo nodo y se añadirá al GLSR.

Durante la identificación de nodos, esto equivale a aplicar el algoritmo del apartado 4.6, en lugar de para todos los nodos del GLSR, sólo para aquellos conectados a un pequeño nivel de profundidad con el último LSR visitado.

5.6 Accesibilidad

El problema de la accesibilidad plantea si una posición es directamente alcanzable desde otra, es decir, si podemos acceder desde la primera a la segunda utilizando tan sólo el navegador reactivo. En general, la única manera de saber si una posición es accesible es comprobar si el navegador tiene éxito al intentar llegar a ella, pero en ciertas condiciones especiales podemos asegurar que una posición será accesible desde otra. Disponer de una medida de la accesibilidad de un nodo tiene varias utilidades en nuestro sistema: saber si podemos llegar a la meta desde la posición actual, saber si un LSR sería alcanzable desde otro, etc.

5.6.1 Visibilidad

Un modo muy sencillo de definir la accesibilidad de un lugar es utilizando la idea de visibilidad: un lugar es accesible desde otro cuando podemos verlo. Para ello utilizaremos la

información suministrada por los sensores: un lugar es accesible si la medida del sensor cuyo vector de dirección apunta más cerca de la dirección correspondiente es mayor que la distancia que separa ambos lugares (ecuación 5.3).

$$s_{pq} > d(p,q) \Rightarrow A(p,q) \qquad (5.3)$$

donde s_{pq} es la medida del sensor que apunta más cerca de la dirección del segmento que une p con q, y d es la función distancia euclídea. Para determinar s_{pq} puede utilizar una ecuación similar a la 4.1 sustituyendo Q por el número de sensores.

5.6.2 Alcanzabilidad

La visibilidad como medida de accesibilidad es en ocasiones demasiado restrictiva. Si s_{pq} es menor que d puede ser porque q esta demasiado lejos de p en relación con el alcance máximo del sensor. Una definición alternativa de accesibilidad sería considerar que (a) q es visible o (b) la visibilidad es la máxima posible (el alcance del sensor) y la distancia es menor que cierto umbral (ecuación 5.4). Si se cumple el punto (b) diremos que la posición es *alcanzable*.

$$[s_{pq} > d(p,q)] \vee [s_{pq} \approx s_{max} \wedge d(p,q) < U] \Rightarrow A(p,q) \qquad (5.4)$$

donde s_{max} es el alcance máximo de los sensores y U es un umbral arbitrario.

Si U es igual a s_{max} tenemos el caso anterior. Cuanto mayor sea U, menos conservadores seremos para designar como accesible un lugar que no se ve, y por tanto mayor será la probabilidad de cometer el error de designar como accesible un lugar que no lo es.

5.6.3 Selección de nodos origen y objetivo de una misión

Un problema relacionado con el de la accesibilidad es el de la selección de nodos inicio y objetivo en una misión. Existen diversas posibilidades para realizar esta selección:

1. Un caso trivial es que el usuario quiera ir precisamente a un nodo del GLSR. Si el usuario ha asociado etiquetas a algunos de los nodos del grafo, y en determinado momento invoca una etiqueta, el sistema deberá planificar un camino desde el último

nodo visitado (por ello, en principio, el más cercano) y el nodo identificado por la etiqueta.

2. Un sistema sencillo —y muy utilizado— consiste en calcular como nodo inicial el nodo más cercano a la posición actual del robot, y como nodo objetivo el nodo más cercano a la posición de la meta. La figura 5.7 muestra las regiones de asignación de cada posición del mundo utilizando este método; cada nodo se ha etiquetado con un color diferente.
3. Un último planteamiento sería utilizar los criterios de accesibilidad definidos anteriormente, de modo que el nodo inicio es el nodo accesible más cercano a la posición inicial del robot, y el nodo objetivo es el nodo accesible más cercano a la posición de la meta. La figura 5.8 muestra las regiones de asignación utilizando el método de la accesibilidad.

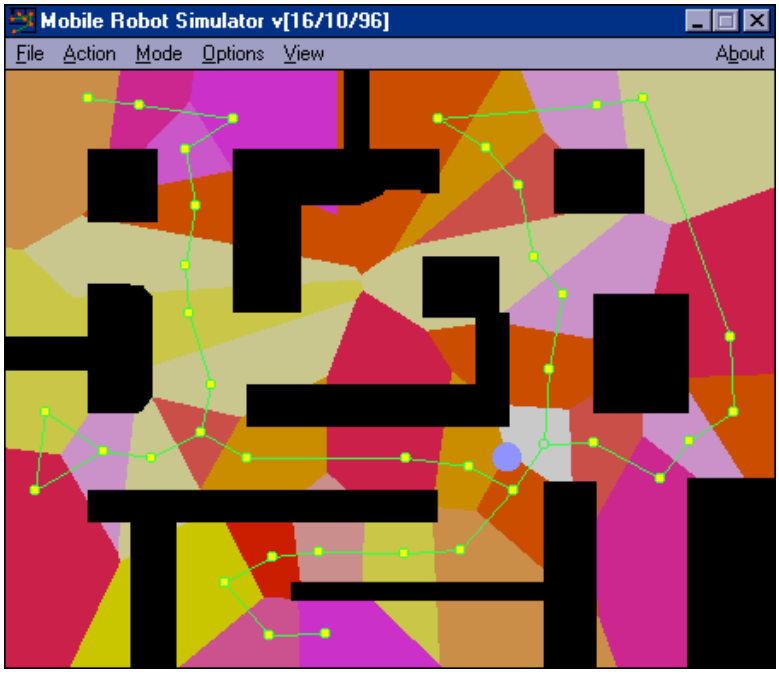


Figura 5.7. Regiones de asignación de nodos según la distancia mínima

Comparando los resultados de los métodos 2 y 3, podemos observar que el método de la distancia mínima asigna regiones a nodos que son claramente inalcanzables desde la posición considerada (ver por ejemplo la zona central de la figura 5.7). El método basado en la accesibilidad, sin embargo, es mucho más preciso realizando la asignación, aunque tiene el inconveniente de que deja regiones sin asignar (en blanco en la figura). Si el robot se

encuentra en la región central del entorno de las figuras 5.7 y 5.8, con el método 2 elegiría un destino erróneo, con el método 3 no obtendría ningún destino.

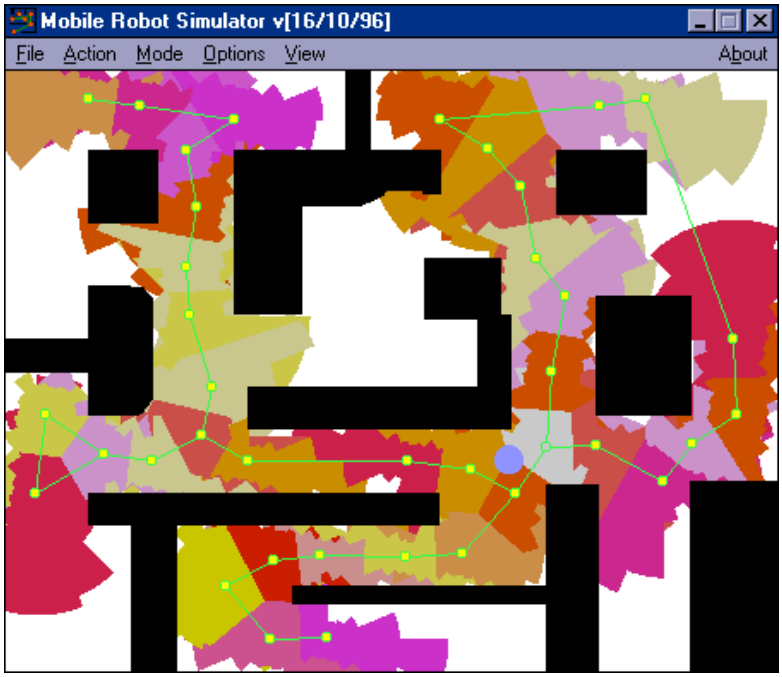


Figura 5.8. Regiones de asignación de nodos según la accesibilidad con U=d_{max}

Dentro de lo malo, es preferible la opción del método 3, porque de no obtener ningún destino podríamos utilizar otro modo de navegación, por ejemplo la exploración basada en cuadrantes, para resolver el problema. Además, esta discrepancia entre métodos es tanto más importante cuanto menos acabado esté el modelo. En etapas tempranas del funcionamiento del sistema es importante saber cuándo podemos utilizar un planificador global y cuándo nos tenemos que contentar con el uso del planificador reactivo, y esto sólo lo permite el método 3, ya que si el modelo no es suficiente para la misión no se identificará ningún nodo.

En lo que respecta exclusivamente a la elección del nodo inicial hay otra posibilidad: el sistema puede realizar algún tipo de exploración hasta identificar algún LSR, y a partir de ahí utilizar el plan. Por ejemplo, en el algoritmo de planificación con sugerencias globales, el robot puede inicializar el rumbo con la dirección del segmento que une la posición del robot y la meta, y avanzar con ese rumbo hasta detectar un LSR. En caso de llegar a un MLS se elige una nueva dirección al azar. Siguiendo este procedimiento podemos asegurar que en algún momento se detectará e identificará algún LSR; a partir de ahí basta con seguir las sugerencias.

5.7 Mantenimiento y replanificación en un GLSR

Un plan puede quedar invalidado por dos motivos: bien porque el sistema detecte inesperadamente[10] un MLS, bien porque el sistema no consigue alcanzar un LSR después de un tiempo prudencial. En ambos casos el problema se debe a los cambios en el entorno. En esta circunstancia el GLSR será modificado incrementando la probabilidad de ocupación del arco implicado e inhibiendo dicho arco temporalmente; tras esto se calculará un plan alternativo. Por otra parte, siempre que se cubra con éxito el trayecto entre dos nodos, la probabilidad de ocupación del arco correspondiente se decrementará.

5.7.1 Actualización bayesiana de los arcos

El teorema de Bayes se utiliza frecuentemente en los procesos de actualización de modelos [Fisher *et al*, 1991; Werbos, 1994; Elfes, 1990; 1991]. Utilizando este teorema hemos derivado un conjunto ecuaciones que permiten revisar la probabilidad de ocupación de un arco según el resultado obtenido en el intento de recorrerlo. Según Bayes, la probabilidad de que se dé un suceso x_k supuesto que se dio un suceso y se define según la ecuación 5.5.

$$p(x_k / y) = \frac{p(y / x_k)\, p(x_k)}{\sum_i p(y / x_i)\, p(x_i)} \qquad (5.5)$$

En nuestro caso sólo existen dos posibles sucesos complementarios: que se tenga éxito atravesando el arco o que se fracase; los llamaremos “éxito” y “fracaso”. El modelo incluye dos probabilidades: la probabilidad de que el arco esté ocupado y la probabilidad de que el arco esté libre. Teniendo en cuenta el suceso “fracaso” y observando que la probabilidad de ocupación y la de no ocupación son complementarias, podemos modelizar nuestro problema con la expresión (5.6).

$$p(oc / \ fracaso) = \frac{p(\ fracaso / oc)\, p(oc)}{p(\ fracaso / oc)\, p(oc) + p(\ fracaso / \neg oc)(1 - p(oc))} \qquad (5.6)$$

[10] Inesperadamente porque si todo va bien no debería encontrarse ningún MLS cuando se esté siguiendo un plan global.

donde *p*(*oc*) es la probabilidad de ocupación a priori, *p*(*oc/fracaso*) es la probabilidad de ocupación a posteriori, *p*(*fracaso/oc*) es la probabilidad de que si el modelo pronostica que el arco está ocupado se dé un fracaso y *p*(*fracaso/¬oc*) es la probabilidad de que si el modelo pronostica que el arco no estará ocupado se dé un fracaso. El cociente entre estos dos últimos (ecuación 5.7) se denomina *razón de verosimilitud*, y representa el valor informacional del dato (*fracaso*) con respecto a la verdad de la hipótesis [Martínez-Arias, 1991]. Cuanto más alto sea este valor, más radicales serán las actualizaciones de la probabilidad de ocupación al darse el suceso fracaso. Cuanto más cercano a 1 sea, más conservadoras serán estas modificaciones. Los valores por debajo de 1 no tienen sentido, ya que provocan una actualización invertida, es decir, decrementan la probabilidad de ocupación cuando se da un fracaso.

$$\gamma_f = \frac{p(\,fracaso \,/\, oc)}{p(\,fracaso \,/\, \neg oc)} \tag{5.7}$$

Teniendo en cuenta la razón de verosimilitud, y llamando p_o^{t+1} a la probabilidad a posteriori y p_o^t a la probabilidad a priori, obtenemos la ecuación 5.8, que indica cómo actualizar la probabilidad de ocupación cuando se da un fracaso.

$$p_o^{t+1} = \frac{p_o^t}{p_o^t + \dfrac{(1 - p_o^t)}{\gamma_f}} \tag{5.8}$$

El planteamiento es idéntico cuando se da el suceso “éxito”, salvo que en este caso la razón de verosimilitud se calcula teniendo en cuenta que lo que favorece al éxito es que el arco esté *no ocupado*. Utilizando el teorema de Bayes para este caso (5.9) y la razón de verosimilitud del éxito (5.10), obtenemos la ecuación 5.11, que indica cómo se modifica la probabilidad de ocupación cuando se da un éxito.

$$p(oc \,/\, exito) = \frac{p(exito \,/\, oc)\, p(oc)}{p(exito \,/\, oc)\, p(oc) + p(exito \,/\, \neg oc)(1 - p(oc))} \tag{5.9}$$

$$\gamma_e = \frac{p(exito \,/\, \neg oc)}{p(exito \,/\, oc)} \tag{5.10}$$

$$p_o^{t+1} = \frac{p_o^t}{p_o^t + \gamma_e(1 - p_o^t)} \qquad (5.11)$$

Una vez fijados γ_f y γ_e disponemos de un procedimiento para actualizar la probabilidad de ocupación de los arcos en sucesivas misiones, sin más que determinar si un arco puede o no recorrerse y aplicar la ecuación 5.8 o 5.11, según el caso. Valores de γ cercanos a 1 producen una evolución conservadora, es decir, los valores de probabilidad se actualizarán despacio y se necesitarán muchas ocurrencias del evento correspondiente para la probabilidad de ocupación se modifique sensiblemente. Valores altos de γ indican posturas más radicales.

En la figura 5.9 se presentan varias curvas de actualización para diferentes valores de γ_f y γ_e. Las líneas continuas indican la actualización de p_o^{t+1} frente a p_o^t cuando se da el suceso "fracaso" y las líneas discontinuas indican la actualización cuando se da el suceso "éxito".

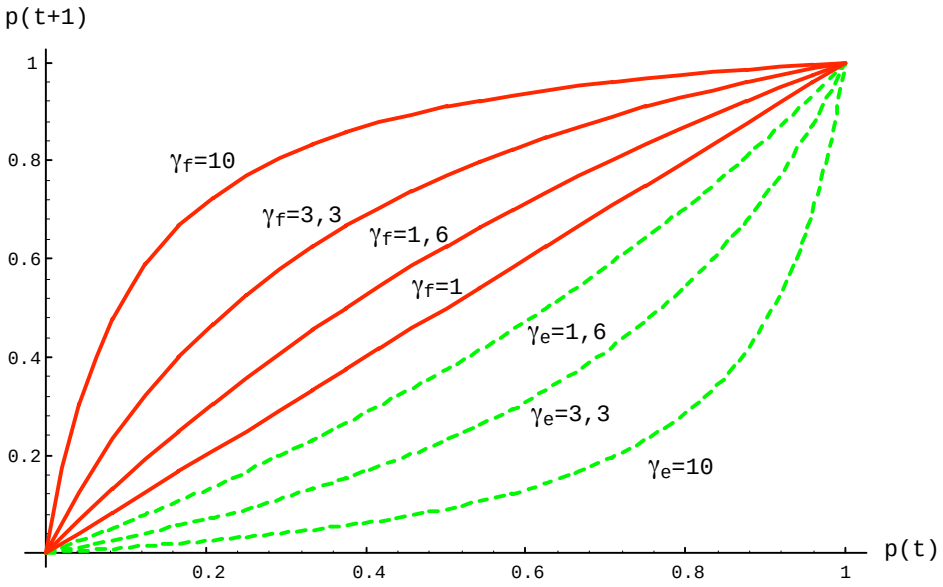


Figura 5.9. curvas de actualización de probabilidad para diferentes γ

Las ecuaciones 5.8 y 5.11 nos proporcionan una probabilidad de ocupación que se almacenará en cada arco. A partir de esta probabilidad y del coste del arco (medido en distancia o en tiempo de recorrido entre los nodos que conecta) podemos calcular el *coste modificado* del arco utilizando la ecuación 5.12. Este coste modificado será el que utilicen los algoritmos de planificación. La idea subyacente es que el coste de un arco deberá incrementarse en función de su probabilidad de ocupación. Cuando la probabilidad de

ocupación sea 0, el coste modificado será el coste original. Cuando la probabilidad de ocupación tienda a 1, el coste modificado deberá tender a infinito.

$$C_{\text{mod}} = \frac{C}{1 - p_o} \qquad (5.12)$$

donde C es el coste original del arco, C_{mod} es el coste modificado y p_o es la probabilidad de ocupación actual del nodo.

Esta función de coste modificado es interesante porque nos da la posibilidad de considerar el coste de un camino como una combinación entre su longitud (en tiempo o en espacio) y su probabilidad de ocupación. Sin embargo, hay que resaltar que la función de coste modificado *no cumple la restricción de la monotonía*, hecho que tiene algunas implicaciones importantes en los procesos de planificación. Con algunas precauciones, tanto A* como el algoritmo de búsqueda en oleadas propuesto en esta tesis garantizan encontrar la solución óptima aún en estas circunstancias.

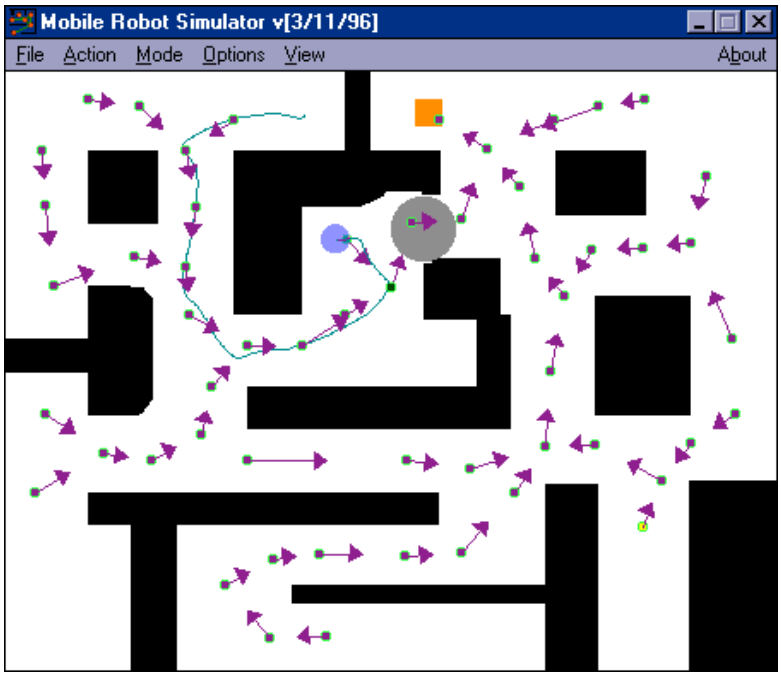


Figura 5.10. Plan invalidado por un nuevo obstáculo en el entorno

En las figuras 5.10 y 5.11 mostramos un ejemplo de replanificación. En la figura 5.10 se ve el plan inicial, que no puede seguirse hasta el final debido al obstáculo; en la figura 5.11 se ha inhabilitado el arco correspondiente y se ha recalculado el plan. Del mismo modo, aunque esto

no se aprecia en las figuras, la probabilidad de ocupación de los arcos del grafo ha sido actualizada, castigando el arco obstruido y premiando los que han tenido éxito.

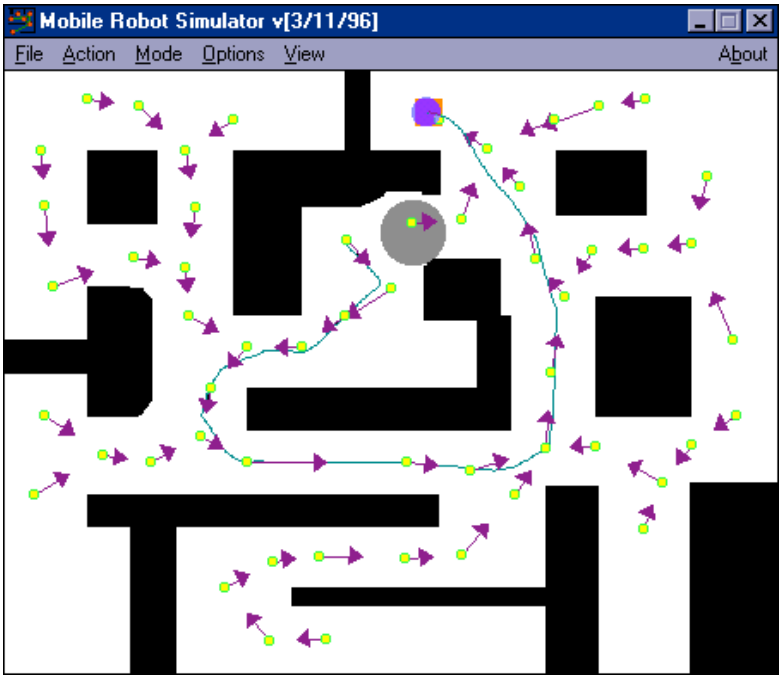


Figura 5.11. Plan recalculado para la nueva situación

5.7.2 Reflexión sobre las razones de verosimilitud

Hemos visto que la razón de verosimilitud regula cómo afecta un suceso a la actualización de la probabilidad de ocupación de los arcos. Valores altos dan mucha credibilidad al dato percibido y por ello modifican fuertemente dicha probabilidad, mientras que valores cercanos a 1 son más conservadores. Si utilizamos razones γ_f y γ_e iguales, dos sucesos contrarios se cancelan. Es decir, si se da un suceso "fracaso" seguido de un suceso "éxito", la probabilidad final es la que teníamos antes de los dos sucesos.

Como se vio en el capítulo 4, los arcos del GLSR se construyen utilizando un procedimiento altamente robusto. Este sistema de construcción, junto con la planificación en oleadas, hacen que sea mucho más frecuente que un arco esté habilitado (es decir, que se dé un éxito) a que esté inhabilitado (es decir, que se dé un fracaso). Por ello los arcos se premiarán muchas veces (con la ecuación 5.11) y se castigarán muy pocas (con la ecuación 5.8). Si queremos que un eventual castigo tenga efectos apreciables será, pues, necesario que

los castigos sean más intensos que los premios[11]. Para obtener castigos más intensos es suficiente utilizar valores de γ_e mayores que los de γ_f.

5.7.3 Memoria del Mapa Cognitivo y memoria de misiones

En la memoria a largo plazo pueden almacenarse las sugerencias o direcciones recomendadas de los nodos con respecto a ciertas misiones ejecutadas frecuentemente, lo que evita tener que calcularlas cada vez. Es como lo que nos sucede cuando recorremos un trayecto habitual; no necesitamos planificarlo de antemano, simplemente aplicamos en ciertos puntos decisiones preestablecidas. Es frecuente incluso tomar la decisión preestablecida cuando el objetivo no es el habitual, con el consiguiente error; hasta tal punto son fuertes las sugerencias de los lugares que se visitan a menudo en trayectos hacia el mismo objetivo.

Este hecho aclara el papel del aprendizaje por refuerzo en el contexto de la planificación de caminos. El aprendizaje por refuerzo es dependiente del objetivo, y por tanto construye modelos poco potentes, que sólo son útiles para un conjunto reducido de objetivos. No obstante, cuando un mismo problema se resuelve innumerables veces la planificación va dando paso a la memoria, hasta que finalmente no se planifica sobre la memoria del mapa cognitivo, sólo se recupera la solución ya almacenada en la memoria de misiones.

5.8 Resultados

A lo largo de este capítulo hemos mostrado tres mecanismos de utilización de la información almacenada en el mapa cognitivo:

- Una planificación que utiliza los cuadrantes de exploración. Esta planificación es local —sólo utiliza información del último LSR detectado— y basada en sugerencias.
- Una planificación global clásica basada en el algoritmo A*.
- Una planificación global basada en sugerencias. Este tipo de planificación utiliza la búsqueda en oleadas para calcular las sugerencias óptimas de cada LSR existente en el modelo.

[11] Ignoramos si esta conclusión tiene repercusiones pedagógicas.

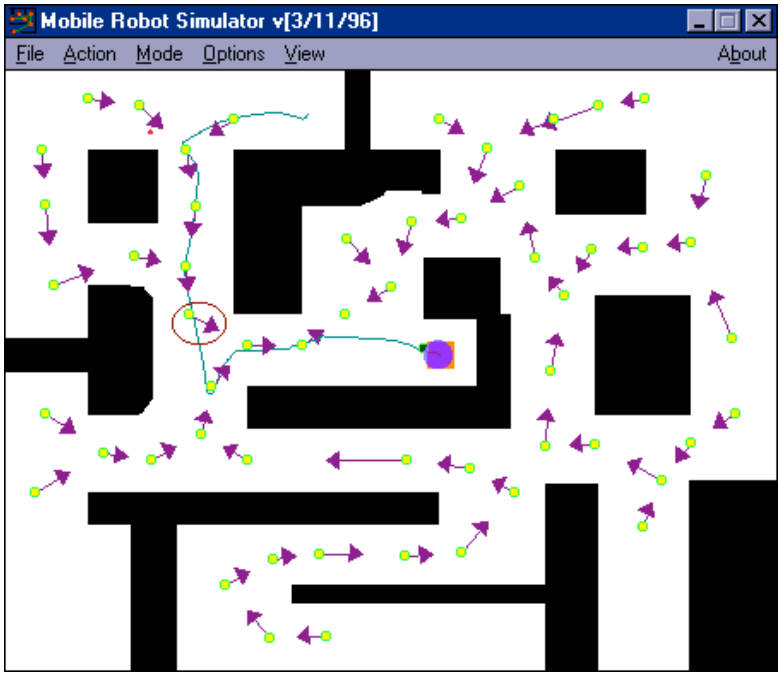


Figura 5.12. Camino obtenido con la planificación en oleadas tras un fallo en la detección de un LSR. Encerrado en un círculo se muestra el LSR no detectado.

Comparando estos tres mecanismos podemos obtener una serie de conclusiones:

- La planificación en oleadas es mucho más robusta que la planificación con A*. En la segunda, si un LSR no se detecta al aplicar un plan, el sistema se pierde. Al llegar a un mínimo local, la misión debería replanificarse. En la planificación en oleadas, si un LSR no es detectado el resultado es sólo una pequeña degradación en la trayectoria, porque el siguiente LSR detectado reconducirá al robot por el camino óptimo (ver figura 5.12). La detección de LSR es tan robusta que, para generar el ejemplo de la figura, el código del simulador tuvo que ser modificado con objeto de obligar a que el LSR encerrado en un circulo no fuera detectado. En el mundo real, la no detección podría originarse por modificaciones en el entorno.
- Los dos planificadores globales disponen de la capacidad de replanificar si algo falla, cosa que se detecta mediante la aparición de algún MLS durante la misión. El planificador local no necesita esta capacidad, ya que va generando el plan *on-line* a medida que el robot se mueve. En el planificador local *generación del plan* y *aplicación del plan* no están separadas: el plan se aplica a medida que se genera.
- La planificación local es menos eficiente que las globales, pero es extremadamente útil cuando no se dispone de modelo del entorno, ya que no sólo permite encontrar

una solución en ausencia de modelo, sino que además es capaz de construirlo durante la misión.

- Los planificadores globales sólo refinan el modelo previo (castigando o premiando arcos), el planificador local añade nuevos nodos en las regiones que no habían sido exploradas. Los planificadores globales utilizan los arcos; el planificador local utiliza los cuadrantes.

Otras aportaciones de este capítulo han sido:

- Introducir la distinción entre *sugerencias globales* y *sugerencias locales*, dependiendo de si las indicaciones dadas en un nodo garantizan que, de ser obedecidas, conducirán o no a la obtención de una ruta óptima (siempre y cuando el entorno no haya sido modificado).
- Hemos propuesto un sistema de acceso rápido a nodos, la *búsqueda semántica*, inspirado en el concepto de red semántica y la memoria a corto plazo del robot.
- Hemos analizado el problema de la accesibilidad de lugares, definiendo un algoritmo que utiliza las ideas de visibilidad y alcanzabilidad entre lugares.
- Se ha propuesto un sistema para la elección de los nodos inicial y final en una misión, utilizando la idea de accesibilidad. Se ha demostrado que este sistema mejora la opción de seleccionar simplemente el nodo más cercano.
- Se han proporcionado una serie de expresiones matemáticas, deducidas del teorema de Bayes, que permiten la actualización del coste de los arcos en función del éxito o el fracaso de las tareas propuestas por los planificadores. Así mismo, se ha definido una ecuación que integra el coste original de un arco con su probabilidad de ocupación, proporcionando un *coste modificado*.
- Por último, hemos realizado una serie de reflexiones acerca del parámetro γ, que regula el conservadurismo en la actualización del modelo, y hemos propuesto la posibilidad de almacenar las sugerencias de los nodos para objetivos utilizados con frecuencia, de modo que no sería necesario recalcularlas cada vez.

A lo largo del capítulo se han mostrado una serie de resultados reales de ejecución del sistema en el entorno de simulación, que ilustran los algoritmos y ecuaciones desarrollados.

6. Arquitectura del sistema

En este capítulo resumimos la arquitectura del sistema y sus niveles de funcionamiento. Se describirán los diferentes procesos que operan sobre los contenidos de la memoria y cómo estos procesos, en conjunto, controlan el robot. Analizaremos también los distintos tipos de memoria presentes en el diseño del sistema, sus contenidos y su significado. Finalmente, estudiaremos cómo esta arquitectura cognitiva, junto con el conjunto de técnicas desarrolladas en la tesis, pueden utilizarse para una aplicación concreta: el control autónomo o semi-autónomo de una silla de ruedas.

6.1 La importancia de la memoria en los procesos cognitivos

La memoria desempeña un papel crítico en nuestro funcionamiento. Tanto en procesos creativos profundos como en procesos triviales, necesitamos un sistema activo de memoria que pueda guiar nuestras acciones y registrar nuestros logros [Lindsay & Norman, 1983].

Es muy importante la noción de *sistema activo* de memoria. Este carácter activo radica en su capacidad asociativa; en esto la memoria humana y la memoria típica de los ordenadores —esencialmente *pasiva*— son muy diferentes.

Como ya adelantamos en el capítulo 1, la psicología se refiere a tres tipos básicos de memoria:

- El *almacén de información sensorial (o imagen sensorial),* que es el registro que un órgano sensorial deja en la mente durante algunas décimas de segundo. Esta memoria mantiene una imagen detallada de la información captada por el sistema sensorial.
- La *memoria a corto plazo*, registro de corta duración (del orden de segundos) que sirve como memoria de trabajo para los procesos cognitivos. Aquí la información *ya está codificada*, categorizada por los mecanismos de reconocimiento de formas.
- La *memoria a largo plazo*, en la que se mantienen registros permanentes de las experiencias vividas, y tiene una capacidad esencialmente ilimitada. En ella es donde reside el verdadero conocimiento de un ser humano.

Otras partes del sistema de la memoria se ocupan del control: de la selección y supervisión generales de las operaciones de la memoria. Los procesos de control regulan cómo se transfiere la información entre los distintos tipos de memoria, y determinan las operaciones que el sistema de memoria ejecuta.

6.2 Arquitectura centrada en la memoria

Nuestro punto de partida a la hora de diseñar la arquitectura del sistema cognitivo del robot móvil ha sido asignarle un papel central a la memoria. A diferencia de la mayor parte de los autores, que dan un papel central a los procesos, para nosotros los procesos se definirán en base a las transformaciones que efectúan sobre diferentes tipos de memoria. Así, por ejemplo, el módulo de planificación se define esencialmente como un proceso que escribe en *la memoria de intención* consignas de movimiento en forma de rumbos deseados de navegación. Para ello lee de la memoria a largo plazo (mapa cognitivo) la información necesaria para determinar qué rumbo debe escribir en cada momento. Por otro lado, el módulo de navegación lee la consigna de rumbo de la memoria de intención, la distribución de las cargas ficticias y la *imagen sensorial* para obtener con ellas un ítem para la *memoria de actuación*: el vector de dirección recomendada que debería seguir el robot.

Este planteamiento es básicamente multitarea en el sentido de que cada proceso es independiente y tiene su propio bucle de funcionamiento; no hay comunicación entre los procesos salvo a través de la modificación de los contenidos de la memoria. Al no haber

dependencias entre procesos, se permite una evolución incremental de la arquitectura; una vez que se han fijado los tipos de memoria básicos, se pueden ir añadiendo procesos de más alto nivel que traduzcan los deseos del usuario en consignas para los niveles más bajos. Del mismo modo se pueden sustituir componentes particulares del sistema siempre y cuando mantengan el mismo interfaz, es decir, utilicen los mismos tipos de memoria.

Los contenidos de la memoria cambian permanentemente. En el caso de la imagen sensorial este cambio se debe a la modificación de las lecturas de los sensores cuando el robot se mueve o cuando se transforma el entorno. En la memoria a corto plazo los cambios se originan por los procesos que elaboran los datos sensoriales o bien por procesos volitivos del robot —esto es, mediante las sugerencias realizadas por el planificador—. En la memoria a largo plazo los cambios se deben a las actualizaciones efectuadas para reflejar las modificaciones del entorno de una misión a otra.

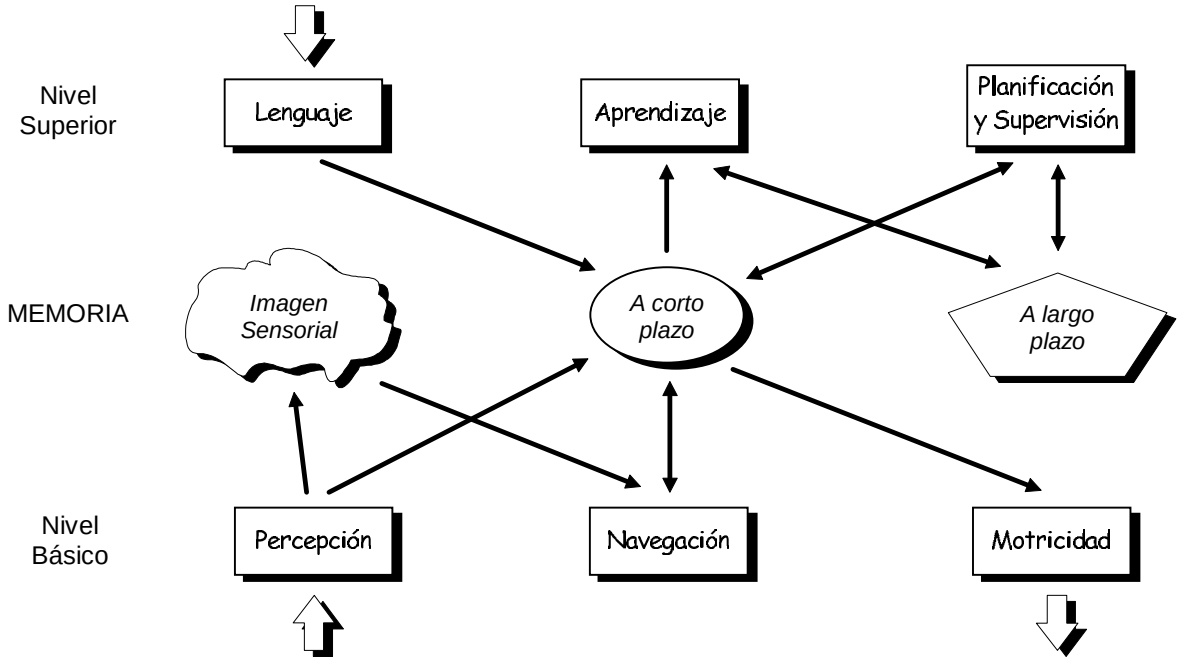


Figura 6.1. Esquema básico de tipos de memoria y procesos. Las cajas rectangulares representan procesos, que hemos organizado en nivel básico y nivel superior.

6.2.1 Almacén de información sensorial

El almacén de información sensorial está compuesto por una imagen sensorial por cada tipo de órgano sensorial del robot. Los procesos de percepción capturan las medidas directamente del *hardware* y, tras un procesamiento más o menos complejo, obtienen una serie de medidas

de rango en determinadas direcciones. En nuestro sistema, además, los procesos perceptivos supervisan la variación temporal de la imagen sensorial (gradiente) con el fin de detectar la aparición de cambios bruscos.

6.2.2 Memoria a largo plazo

La memoria a largo plazo puede tener varias componentes:

- El mapa cognitivo, es decir, el GLSR del robot, que es su núcleo fundamental. Podemos considerar que el GLSR está compuesto de dos secciones: (a) la que almacena los arcos y (b) la que almacena los cuadrantes y el resto de la información. La navegación utilizando sugerencias locales no necesita utilizar la sección (a).
- La memoria de misiones, que es una memoria opcional que almacena por refuerzo las sugerencias utilizadas con frecuencia.

6.2.3 Memoria a corto plazo

La memoria a corto plazo tiene una estructura algo más compleja que las anteriores, dado que disponemos de varios tipos de información que debemos almacenar a corto plazo. Los tipos de memoria a corto plazo que hemos establecido —y que tienen un reflejo más o menos directo en el diseño *software* del sistema— son los siguientes:

1. El objetivo de la misión.
2. La memoria de LSRs, que contiene el último LSR visitado y, si se está detectando alguno, el LSR actual.
3. Una memoria de arcos inhabilitados, que son arcos que se desactivan temporalmente después de un fracaso en el plan actual para no ser tenidos en cuenta en la replanificación.
4. La memoria de MLS, que indica si se está detectando un mínimo local severo.
5. Una memoria para las cargas ficticias (con capacidad para 5 items).
6. Una memoria de intención (rumbo).
7. Una memoria de actuación (vector de velocidad a aplicar).
8. Un plan.

Los procesos que relacionan todos los tipos de memoria se muestran en la figura 6.1. En la figura 6.2 se da una versión en detalle de los componentes de la memoria a corto plazo.

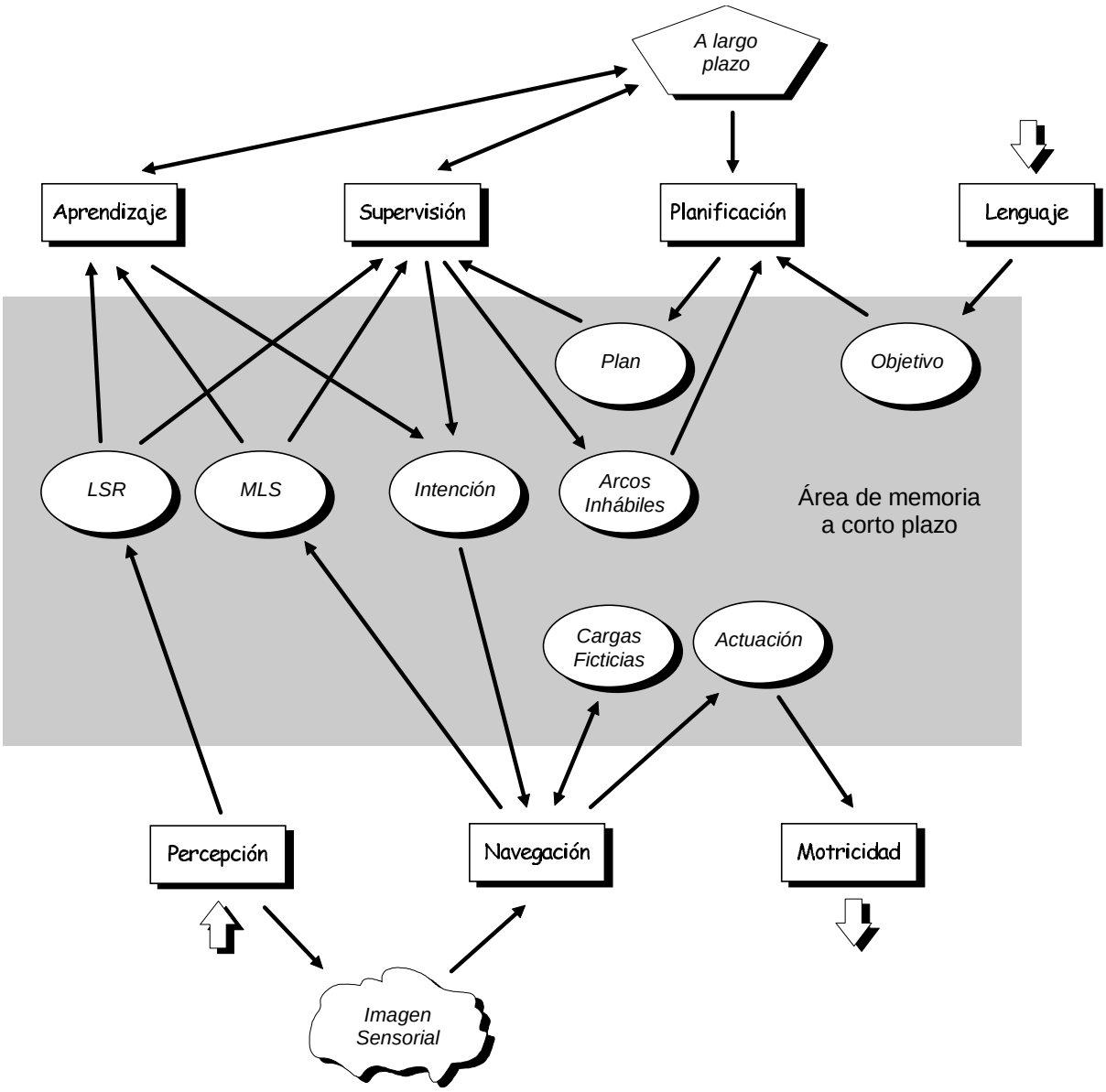


Figura 6.2. Esquema extendido de tipos de memoria y procesos

6.3 Vinculación entre memoria y procesos

Como ya se ha comentado, existe una fuerte relación entre tipos de memoria y procesos que los manipulan. En este punto comentaremos brevemente cada proceso de la figura 6.2 junto con los tipos de memoria que utiliza:

- **Percepción**. Obtiene, a partir de los sensores del sistema, la *imagen sensorial*. Adicionalmente identifica si estamos en un LSR, notificándolo a la *memoria de LSRs*.
- **Navegación**. Se encarga de transformar una consigna de movimiento de *la memoria de intención* en una dirección real de movimiento para la *memoria de actuación*, teniendo en cuenta las limitaciones impuestas por el entorno (*imagen sensorial*). Además de esto se encarga de la gestión de las cargas ficticias y la consiguiente detección de MLSs, accediendo, respectivamente, a la *memoria de cargas ficticias* (lectura/escritura) y a la *memoria de MLS* (escritura).
- **Motricidad**. Lee de la *memoria de actuación* y, atendiendo a la cinemática y a la dinámica del robot, obtiene un comando de movimiento que será enviado al controlador de los motores.
- **Lenguaje**. Obtiene, a través de un interfaz con el usuario, que puede ser de diversos tipos, el objetivo de la misión. Éste es un objetivo general que servirá como entrada a las tareas de planificación.
- **Aprendizaje**. Utilizando la información almacenada en las memorias *de LSR*, *de MLS* y la *memoria a largo plazo*, obtiene nuevos datos para incrementar o actualizar la propia *memoria a largo plazo*. Cuando utilizamos la exploración basada en cuadrantes, ésta proporciona comandos de movimiento que se escriben en la *memoria de intención*.
- **Planificación**. Tomando como entrada la *memoria de objetivo*, la *memoria a largo plazo* y la información de la *memoria de arcos inhabilitados*, obtiene un plan de actuación que se almacena en la *memoria de plan*.
- **Supervisión**. Se corresponde con el ejecutivo de planes de las arquitecturas clásicas. Tomando un plan de la *memoria de plan*, y utilizando la información de la *memoria de LSR*, obtiene y escribe comandos en la *memoria de intención*. Se ocupa, por tanto, de establecer sugerencias al módulo de navegación. Ocasionalmente, si se activa la *memoria de MLS*, añade el arco implicado a la *memoria de arcos inhábiles* y actualiza el modelo de la *memoria a largo plazo*. Cuando la *memoria de arcos inhábiles* cambia, se requiere una nueva planificación.

6.4 Niveles de funcionamiento

Si eliminamos alguno de los tipos de memoria, junto con los procesos que operan sobre ellas, obtenemos niveles de funcionamiento del sistema más simples. Por ejemplo, podemos suponer que no disponemos de memoria a largo plazo y, consecuentemente, no podemos usar el planificador basado en el GLSR. El sistema, no obstante, puede seguir funcionando sin realizar una planificación global del camino. Para conseguirlo, las consignas de movimiento (memoria de intención) que proporcionaba el módulo de supervisión se deberán proporcionar ahora como objetivos directos del sistema. A lo largo del desarrollo de esta tesis hemos definido tres niveles estratégicos de planificación, y cada uno de ellos da lugar a un repertorio distinto de capacidades y uso de la memoria. Todos estos niveles de funcionamiento se resumen en la tabla 6.3. Los tipos de memoria se refieren a los descritos en el apartado 6.2.3.

	Reactiva	LSRs	Modelo cognitivo
Exploración	Rumbo al azar cada vez que se detecta MLS	Exploración basada en cuadrantes	Refinamiento del GLSR utilizando la visibilidad de nodos
Planificación	Navegación con campos de potencial y cargas ficticias	Planificación basada en cuadrantes (sugerencias locales)	a) A* b) Frente de onda
Tipos de memoria a corto plazo	2, 4, 5, 6, 7	1, 2, 4, 5, 6, 7	Todas
Uso de memoria a largo plazo	No	Sí (cuadrantes)	Sí (arcos)

Figura 6.3. Tabla de niveles de funcionamiento

6.5 Aplicación al guiado autónomo de una silla de ruedas

En las aproximaciones tradicionales al diseño de ingeniería se consideran dos categorías de procesos en la integración hombre-máquina: los que realiza el ser humano y los que debe realizar la máquina. En la parte correspondiente al ser humano los procesos constitutivos son *percepción, procesamiento de la información* y *control*, mientras que en la máquina se consideran procesos de *lectura de controles, operación* y *representación de información* [Chapanis, 1976].

En las sillas de ruedas motorizadas tradicionales, el control por parte del usuario requiere una realimentación continua de órdenes, frecuentemente a través del uso de un *joystick*, de manera que la comunicación se establece a un nivel primitivo: el usuario controla directamente los motores de la silla. Este estilo de comunicación, ya de por sí poco ergonómico, resulta particularmente gravoso cuando el usuario tiene reducidas sus capacidades sensoriales o motoras debido a algún tipo de disfunción. Por poner un ejemplo, el control de un *joystick* para un paralítico cerebral resultará una tarea ardua, o incluso imposible, dependiendo del nivel de parálisis que presente.

En lo que sigue se defenderá la necesidad de diseñar sistemas en los que el flujo de información entre usuario y máquina sea lo más bajo posible. Con este objetivo se planteará una remodelación del diseño de sillas de ruedas desde el punto de vista de la cantidad de información que el usuario debe suministrar a la máquina para que ésta satisfaga sus deseos de movilidad. Para ello, e invirtiendo los términos, consideraremos que los procesos de la máquina, en este caso la silla de ruedas, deberán ser los que tradicionalmente se atribuyen al ser humano, es decir, percepción, procesamiento de la información y actuación.

El proceso de percepción será doble: por una parte se obedecerán las demandas de movilidad del usuario —atendidas por el módulo de lenguaje de nuestra arquitectura—, y por otra se dispondrán mecanismos de percepción del entorno que ayuden a la silla a realizar parte de las tareas de control que en los modelos tradicionales se vería obligado a realizar el propio usuario —atendidas en nuestra arquitectura por el módulo de percepción—. Esta descarga de tareas potenciaría en general la ergonomía de cualquier sistema para cualquier usuario, aunque es particularmente útil en el caso de usuarios con algún tipo de discapacidad sensorial, motora o cognitiva.

El nuevo enfoque propuesto presenta sin embargo un inconveniente: si pretendemos que un sistema cognitivo descargue al usuario de las tareas de bajo nivel será necesario desarrollar un conjunto de algoritmos capaz de realizar estas tareas. En definitiva, más ingeniería en el desarrollo generará mayor facilidad de operación en el producto final. El papel que juega esta tesis es precisamente ese: proporcionar un conjunto de algoritmos que hagan viable la descarga de estas tareas de bajo nivel.

6.5.1 Modelo alternativo para el guiado de una silla

Haremos referencia a dos clases de canales, en función del tipo de información que transportan: *canales de estimulación* y *canales de respuesta*. Los canales de estimulación transportan información sensorial desde el entorno hasta el sistema (hombre o máquina). Los canales de respuesta transportan información de actuación desde el sistema al entorno. En un sentido general esta diferenciación no es relevante, aunque tiene sentido en los seres humanos porque los dos tipos de canales involucran actividades muy diferentes.

Según la teoría de la información, la cantidad de información que fluye por un canal puede calcularse utilizando la ecuación 6.1 [McCormick & Sanders, 1982]:

$$C = \frac{\log_2 N}{t} \qquad (6.1)$$

donde N es el número de alternativas, supuestas equiprobables, sobre las que hay que decidir en un intervalo de tiempo t.

La máxima cantidad de información que el ser humano es capaz de transmitir a través de un canal de respuesta, si bien varía con la situación, se estima en torno a los 10 bits/seg. [Singleton, 1971]. Otros estudios han establecido limitaciones para los canales de estimulación y para el procesamiento de la señal [Card *et al*, 1983]. Cuando la cantidad de estímulos excede la capacidad del canal se producen fenómenos de estrés [Conrad, 1951]. Teniendo en cuenta la cantidad de estímulos que viajan por un canal, podremos denominar canales con alta carga a aquellos cuyo flujo de información raya la máxima capacidad (alrededor de 10 bits/seg). Por contra, denominaremos canal con baja carga a aquél cuyo flujo de información no llega al 10% de su capacidad (en torno a 1 bit /seg).

Desde este punto de vista, los diseños tradicionales utilizan canales de alta carga tanto para el *input* del usuario como para las respuestas (figura 6.4). Ha habido, no obstante, algunos intentos anteriores por descargar al usuario de alguna de las tareas de control incorporando mecanismos de seguridad en la silla [Mazo *et al*, 1995] o sistemas de navegación [Madarasz *et al*, 1986], siempre dentro de este esquema tradicional. Aquí proponemos un nuevo planteamiento (ver figura 6.5), donde se sustituyen los canales de alta carga hasta y desde el

usuario por canales de baja carga, incorporando canales de alta carga entre la silla y el entorno.

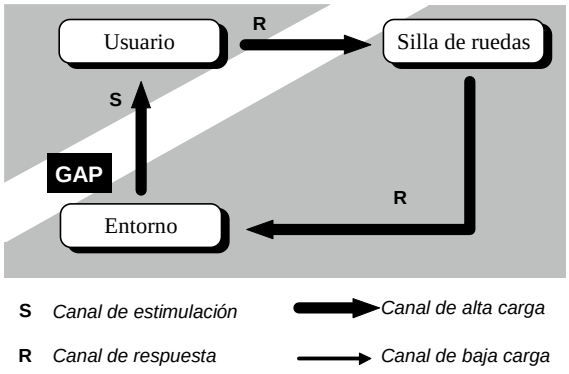


Figura 6.4. Esquema tradicional de la interacción entre usuario, silla y entorno.

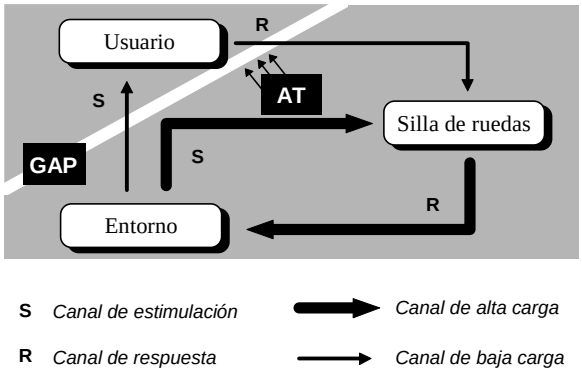


Figura 6.5. Nuevo esquema propuesto para la interacción entre usuario, silla y entorno

Con esta nueva estructura conseguimos liberar al usuario del control directo de la silla, aunque a costa de tener que definir mecanismos que asuman dicho control. Este tipo de mecanismos son el foco de atención de la tecnología asistente (AT en la figura) [Azevedo *et al*, 1994], una nueva disciplina que intenta reducir la distancia (GAP en la figura) entre el discapacitado y su entorno. En las figuras 6.4 y 6.5 puede observarse cómo la sustitución de los canales del alta carga por canales de baja carga permite "aproximar" el entorno al usuario.

6.5.2 Niveles de conducción

Los comandos generados por el usuario responden a deseos de movilidad de éste, y pueden ser expresados en tres niveles de conducción:

1. **Nivel de alta carga**, que equivale al guiado tradicional, es decir, se establece una realimentación permanente entre usuario y silla, a través de un *joystick* o algún otro dispositivo. El flujo de información sobre el canal usuario-silla es muy alto, del orden de el máximo calculado por Singleton, unos 10 bit/seg. En ciertos niveles de deficiencia motora esta solución es inviable, y en cualquier caso el procedimiento es poco práctico.
2. **Nivel de carga media**, en el cual el usuario indica a la silla sus deseos de movilidad en términos de dirección deseada de movimiento, deseos que se introducen directamente en la *memoria de intención*. En este planteamiento, el usuario

constituye la parte volitiva del sistema, mientras que se transfieren a la silla los comportamientos reflejos, utilizando los mecanismos descritos en el capítulo 2. Estimando que el usuario deba comunicar una decisión de entre unas 4 posibles (derecha, izquierda, volver, parar) digamos cada 10 segundos (cada vez que se presenta una alternativa), la carga del canal será, según la ecuación 6.1, de unos 2/10=0.2 bits/seg.

3. **Nivel de baja carga**, en el cual el usuario simplemente indica el lugar al que quiere ir y el sistema cognitivo de la silla se ocupa de planificar y recorrer el camino de modo autónomo, apoyándose en los mecanismos de planificación expuestos en el capítulo 5. Podemos estimar la carga del canal suponiendo una elección, cada 5 minutos, sobre 128 lugares conocidos por la silla, es decir, alrededor de 7/300=0.023 bits/seg. Para poder utilizar este nivel se requiere, lógicamente, (a) un conocimiento previo del entorno por parte del sistema (el GLSR) y (b) una identificación común para usuario y silla de los lugares de interés.

Para adquirir el conocimiento previo del entorno podemos utilizar la exploración basada en cuadrantes, explicada en el capítulo 4, o trabajar durante un tiempo en el nivel de carga media, ya que el modelo se va construyendo permanentemente mientras el sistema de navegación está en funcionamiento.

Para completar este diseño, sería necesario proporcionar al usuario una manera de introducir “etiquetas” que identificaran lugares de interés en un lenguaje común entre usuario y silla. Una posibilidad sería permitir la introducción de estas etiquetas a través de la voz, de modo que, cuando posteriormente la etiqueta fuese evocada, la silla identificase el lugar y generase una misión hacia el lugar correspondiente. La introducción de etiquetas podría efectuarse, de modo alternativo, a través de algún dispositivo mecánico.

6.6 Resultados

En este capítulo hemos expuesto la arquitectura del sistema y sus posibles aplicaciones al guiado autónomo o semi-autónomo de una silla de ruedas.

En lo que se refiere a la arquitectura, esta tesis aporta un enfoque de diseño centrado en la relevancia del papel de la memoria en los procesos cognitivos. Siguiendo esta línea, se han caracterizado los diferentes tipos de memoria de los que dispone el sistema, describiendo cómo son utilizados por los distintos módulos detallados en los capítulos anteriores. También, a modo de resumen, se han estructurado y organizado los algoritmos desarrollados en esta tesis en tres grandes bloques: el nivel de navegación, el nivel de cuadrantes o de planificación reactiva y el nivel de GLSR o de planificación global.

Finalmente, hemos intentado establecer un diseño preliminar de una aplicación del conjunto de técnicas que aporta esta tesis al problema del guiado autónomo de una silla de ruedas, proponiendo un nuevo esquema de comunicación entre usuario y máquina dirigido por criterios de ergonomía.

7. Diseño *software* e implementación

En este capítulo se analiza el diseño *software* realizado. Comenzaremos fijando unos objetivos de diseño que justificarán algunas decisiones sobre la exposición posterior, para pasar a describir las categorías identificadas y las particularidades de implementación relativas a la plataforma NOMAD-200. Hemos utilizado la metodología de diseño orientado a objetos y la notación propuesta por Booch [Booch, 1994]. El lector puede encontrar más información en el apéndice I, que incluye los prototipos de todas las clases implementadas.

7.1 Objetivos del diseño

El objetivo fundamental de nuestro diseño es la construcción de un sistema por capas, de tal modo que las clases cercanas al núcleo sean suficientemente generales como para ser utilizadas, *sin ninguna modificación*, tanto en el robot real como en el simulador. Esto es sumamente importante, porque así nos aseguramos de que el código depurado en el simulador funcionará directamente en el robot real, lo que simplificará la fase de integración.

Un segundo objetivo será descomponer a su vez las operaciones de control del robot en dos capas: una para el control reactivo y otra para todos los métodos relacionados con la construcción y explotación del mapa cognitivo. De este modo el control reactivo basado en el método del potencial puede ser (a) utilizado directamente sin planificación, (b) incorporado a otros sistemas de planificación, y (c) sustituido por otro controlador reactivo.

En tercer lugar, todas las operaciones relativas a la manipulación de los LSRs y del GLSR se encontrarán en clases separadas; esto hace que los procesos de modelado sean independientes de la navegación y clarifica cuándo se accede al uso de procesos cognitivos.

Por último, los módulos de entrenamiento y gestión de mapas autoorganizados y de búsqueda en espacios de estados se han implementado separadamente. Son, en efecto, de carácter general y reutilizables en cualquier otra aplicación.

En la figura 7.1 se muestra un diagrama con las diferentes capas desarrolladas y sus relaciones. La capa 0 corresponde al controlador reactivo; la capa 1 añade a la capa 0 habilidades de gestión de mapas cognitivos; la capa 2a se identifica con el simulador desarrollado y la capa 2b con el programa que controla el robot NOMAD.

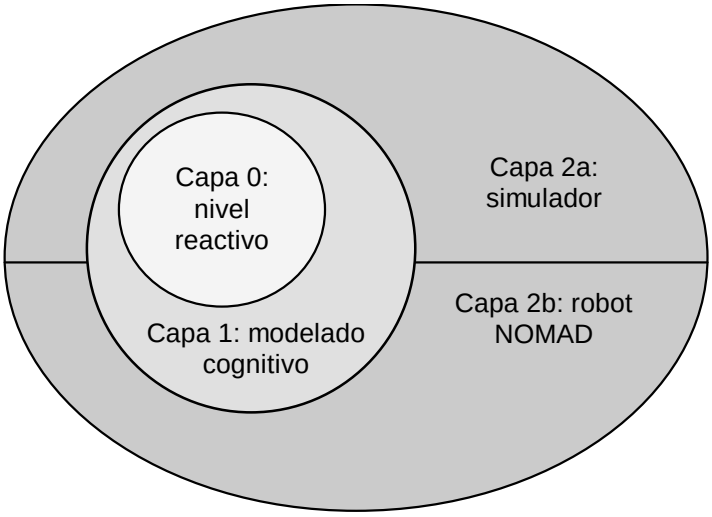


Figura 7.1. Capas del sistema

7.2 Jerarquía de clases

En la figura 7.2 se muestra el esquema general de la jerarquía de clases. Las flechas indican relaciones de herencia; la flecha apunta desde la clase que hereda a la clase heredada. Las uniones con círculo indican relaciones de uso; el círculo se encuentra junto a la clase que usa, es decir, que declara una o más copias del objeto usado. En el texto de Booch se exponen estos y otros elementos que aparecen en los diagramas.

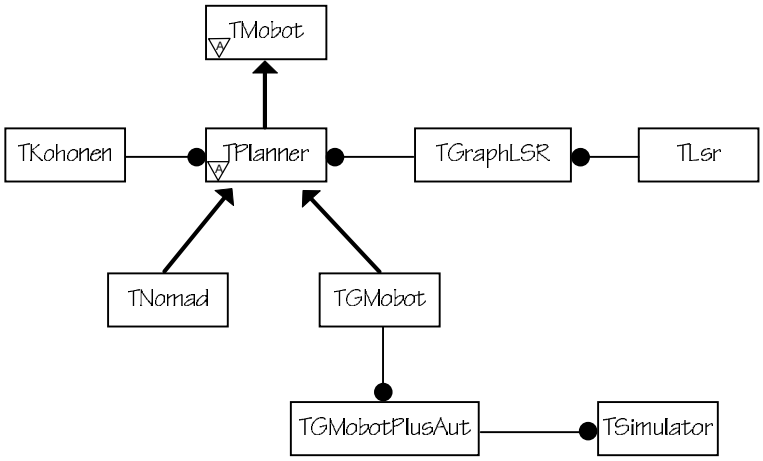


Figura 7.2. Esquema general de diseño

Explicamos a continuación cada una de las clases de la figura 7.2:

- Clase TMOBOT: es una clase abstracta que define los métodos de navegación utilizando el procedimiento del potencial artificial con cargas ficticias expuesto en el capítulo 2.
- Clase TGRAPHLSR: define las estructuras para albergar el GLSR y sus operaciones básicas.
- Clase TLSR: define los métodos y estructuras para crear y manipular LSRs.
- Clase TKOHONEN: implementa los procesos de entrenamiento y propagación para redes de Kohonen o mapas autoorganizados.
- Clase TPLANNER: es una clase abstracta que define los métodos de construcción y planificación expuestos en los capítulos 3, 4 y 5.
- Clase TNOMAD: define los métodos particulares para el robot NOMAD-200. Básicamente incluye el nivel de pilotaje, que se comentará en este mismo capítulo, un pequeño interfaz para comunicar instrucciones al robot y una serie de enlaces con las instrucciones de la biblioteca de comandos del robot. Estos enlaces traducen las instrucciones de TMOBOT y de TPLANNER a instrucciones de la biblioteca: lectura de los sensores, envío de comandos de movimiento, etc.
- Clase TGMOBOT: define los mecanismos necesarios para operar con un robot gráfico simulado en la pantalla del ordenador.

- Clase TGMOBOTPLUSAUT: Incorpora las estructuras necesarias para operar con varios robots gráficos del tipo TGMobot.
- Clase TSIMULATOR: Define los métodos del interfaz del simulador.

La figura 7.3 muestra un esquema más detallado de la jerarquía de clases, señalando la capa a la que pertenece cada clase. En él hemos incluido todas las clases implementadas y sus relaciones. Comentamos a continuación las clases que no aparecían en la figura 7.2:

- Clases MATRIZ y VECTOR: implementan las operaciones básicas sobre matrices y vectores en memoria dinámica (*heap*). Son utilizadas por la clase TKOHONEN, ya que para el entrenamiento de redes de neuronas se requiere el uso de matrices de grandes dimensiones.
- Clase TEJEMPLOS: define las operaciones básicas para manipular ejemplos de entrenamiento para redes de neuronas.
- Clase TCART: implementa una serie de operaciones para trabajar con coordenadas cartesianas y polares, así como mecanismos de conversión entre ellas.
- Clase TCHARGES: define una estructura capaz de almacenar y manipular cargas ficticias.
- Clase TCRONO: contiene métodos de gestión de tiempos. En particular, permite cronometrar lapsos de tiempo entre eventos. Se usa para calcular la duración del ciclo de control en el robot NOMAD.
- Clase TSENSOR: permite representar las características de un sensor.
- Clase TSENSORYSYSTEM: define el sistema sensorial de un robot.
- Clase TSENSORYMAP: contendrá el registro de las lecturas de un sistema sensorial.
- Clases TCONS y TARC: se usan para definir listas de LSRs.
- Clase TSPACE: define métodos generales de exploración en espacios de estado.
- Clase TSPACELSR: particulariza la clase TSPACE para su uso en exploración de GLSRs.
- Clase TWORLD: contiene los objetos del mundo cargado en el simulador.

- Varias clases de ObjectWindows Library: son un conjunto de clases proporcionadas por Borland para el desarrollo de interfaces en Windows 95.

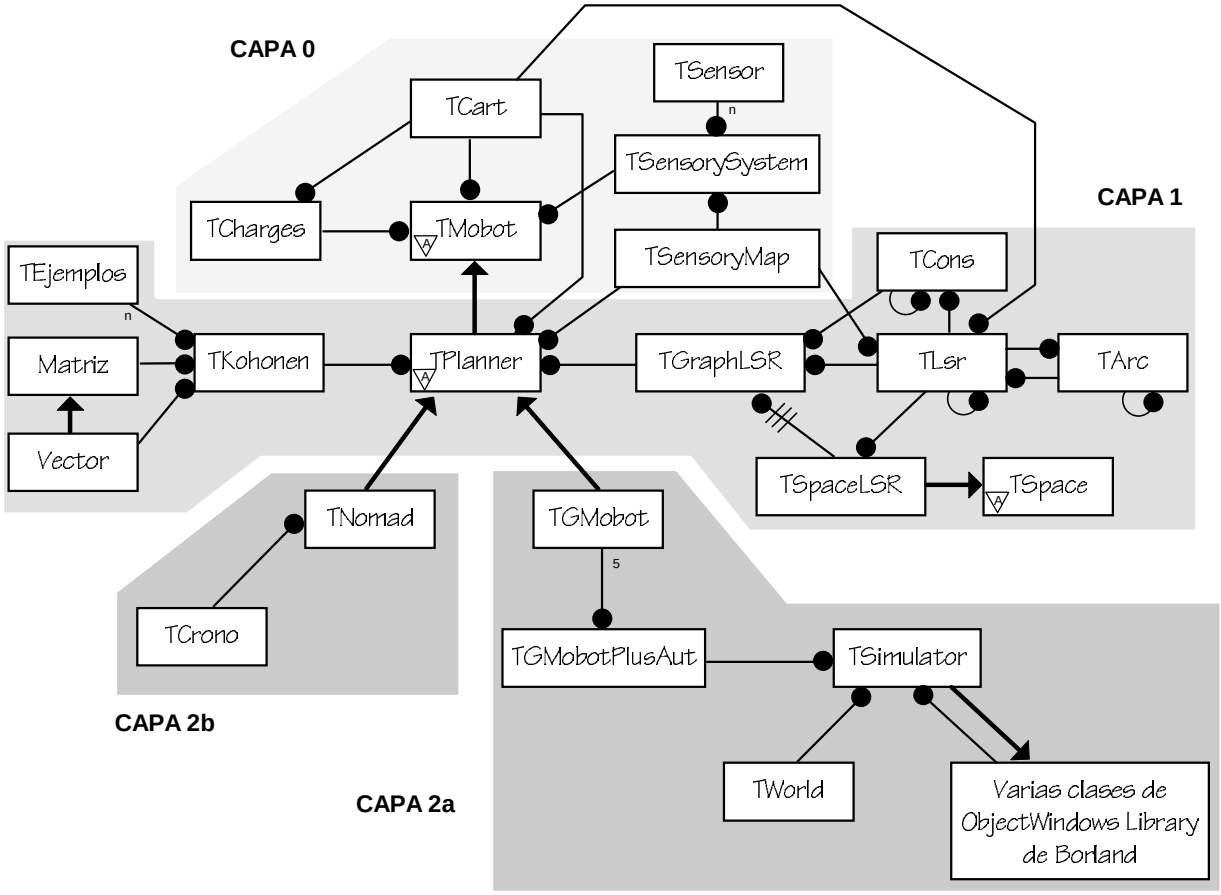


Figura 7.2. Esquema completo de diseño: jerarquía de clases y capas

7.3 Simulador

El simulador proporciona un interfaz para probar todas las teorías y modelos desarrollados en esta tesis, implementados en las clases de las capas básicas (capas 0 y 1). Con este programa podemos elegir entre los distintos modos de funcionamiento (planificadores, métodos de identificación de LSRs, umbrales, etc.) y, fijando una meta para el robot, comprobar en tiempo real la trayectoria que el sistema describiría.

Por su simplicidad de uso no entraremos en detalles acerca de su manejo. Sólo remarcaremos que todos los ejemplos mostrados a lo largo de esta tesis corresponden a salidas reales del simulador.

7.4 Particularidades introducidas para el robot Nomad 200

En este punto comentaremos las peculiaridades de la plataforma NOMAD-200, utilizada para las pruebas experimentales. La clase TNOMAD tiene en cuenta estas peculiaridades y define los mecanismos necesarios para enlazar el resto de las clases con la biblioteca proporcionada por el fabricante del robot.

7.4.1 Funciones de la biblioteca del fabricante

Aunque no vamos a entrar en detalle en este tema —para ello nos remitimos al manual del robot—, queremos dar una idea somera de las herramientas que proporciona el fabricante para actuar sobre la plataforma. Con ellas podremos dar soporte a los mecanismos de pilotaje, navegación y planificación desarrollados en la tesis. Estas herramientas hacen posible:

- La lectura de los sensores del robot, a través del acceso a un *array* global que contiene, en pulgadas, la distancia medida en la dirección del sensor.
- El envío de instrucciones de velocidad a los motores.
- El acceso a los sistemas de estimación odométrica y goniómetrica, que permiten determinar, con una precisión no demasiado elevada, la posición y orientación de la plataforma.
- Algunas funciones auxiliares, como situar al robot en su origen de coordenadas angulares (*hardhome*), establecimiento de las aceleraciones que usará el robot para alcanzar las velocidades que se le indiquen, fijación del orden de disparo de los sensores de ultrasonidos, etc.

El *software* del fabricante incluye algunas otras funciones, tales como las de comunicación por el puerto serie, desplazamiento a una posición fija, etc., que no han sido utilizadas en nuestro desarrollo.

7.4.2 Sensores disponibles en la plataforma

Los modelos propuestos en este trabajo permiten la utilización de una variada gama de sensores: ultrasonidos, infrarrojos, sensores de tacto, medidores láser de rango e incluso visión artificial. El fabricante comercializa todos estos sistemas sensoriales. Por el momento, sin

embargo, sólo disponemos del sistema sensorial basado en ultrasonidos (o sónar), de modo que todas las pruebas experimentales utilizan dicho sistema.

El sistema sensorial basado en ultrasonidos incorporado en la plataforma NOMAD-200 incluye 16 sensores Polaroid que conforman un cinturón alrededor del robot. Los 16 sensores están equiespaciados (uno cada 22,5 grados) y aproximadamente a 80 cm. del suelo.

Es bien conocido que los ultrasonidos —si bien constituyen un sistema barato y relativamente fiable— presentan algunas carencias, que no vamos a enumerar aquí. Desde un punto de vista práctico, la más grave de todas es que son "ciegos" a objetos demasiado pequeños. Al margen de esto, todos los sensores del cinturón de ultrasonidos están situados a la misma altura, de modo que objetos demasiado bajos o demasiado altos tampoco son detectados. En particular, las patas de las sillas, papeleras y tableros de ciertas mesas son invisibles para nuestro sistema sensorial, lo que en ciertas situaciones puede suponer un peligro para la plataforma. Hamos detectado también que las ventanas con grandes cristaleras no reflejan bien los ultrasonidos.

Para suplir estas carencias, que en ningún caso se deben a los mecanismos desarrollados sino a problemas del sistema sensorial, sería conveniente incorporar a la plataforma, como mínimo, sensores táctiles e infrarrojos. En caso de discrepancia entre las medidas de varios sensores que apuntan en la misma dirección, es una buena norma utilizar la menor de todas ellas.

Afortunadamente, salvo en habitaciones estrechas con muchos objetos de los tipos descritos como conflictivos, los ultrasonidos no presentan graves problemas. En particular, como podrá observarse en el capítulo dedicado a pruebas experimentales, el comportamiento en los pasillos de la facultad ha sido aceptable.

7.4.3 Cinemática del robot

La plataforma dispone de 3 grados de libertad, dos de rotación (uno para las ruedas y otro para la torreta) y uno de traslación. La rotación de la torreta permite orientar los sensores hacia distintos lugares. Esto es útil, especialmente, para medidores láser o cámaras de visión. En nuestro caso, como los ultrasonidos están distribuidos uniformemente a lo largo del perímetro

del robot, no necesitaremos rotar la torreta excepto —como se comentará en el capítulo siguiente— para propósitos de recalibrado.

Esta configuración en rotación y traslación hace que no pueda transferirse instantáneamente al robot una determinada velocidad de movimiento $\vec{v}$, especialmente en lo que respecta a la dirección de movimiento. En lugar de esto, habrá que comunicar una determinada dirección de rotación hasta que las ruedas alcancen la dirección deseada, y una vez fijada la dirección deseada, avanzar.

Sin embargo, este procedimiento no es práctico, porque si la dirección deseada no se aparta demasiado de la actual no tiene sentido parar la translación hasta que la orientación deseada se alcance; como solución se propone regular la velocidad de translación en función del error cometido en la orientación. Mientras mayor el error, menor la velocidad de translación, y viceversa. Este problema se tratará en detalle en el apartado siguiente.

7.4.4 Módulo de motricidad o de pilotaje

A lo largo de este trabajo hemos obviado los problemas relacionados con la dinámica del robot. Pero un robot es un sistema físico con una masa determinada; sus motores tienen limitaciones en la fuerza máxima que pueden desarrollar, y por tanto las aceleraciones y deceleraciones no pueden ser de magnitud infinita. La cinemática, como acabamos de comentar en el punto anterior, también introduce limitaciones en el tiempo necesario para conseguir que el robot adquiera un determinado rumbo.

Para tener en cuenta estas restricciones físicas es necesario introducir un nivel de control, que adecue las intenciones de los algoritmos de navegación a la realidad de los motores. Por poner un ejemplo extremo, supóngase que el robot se dirige en una determinada dirección a cierta velocidad $\vec{v}$ determinada por una consigna de movimiento $\vec{F}$ y un obstáculo móvil (por ejemplo una persona) se sitúa a gran velocidad ante él. Supongamos que este hecho provoca que el algoritmo de navegación calcule una nueva consigna $-\vec{F}$ opuesta a la anterior. Es evidente que no es posible para un objeto físico cambiar instantáneamente la velocidad de $\vec{v}$ a $-\vec{v}$. Por otro lado, y puesto que el robot se controla en translación y rotación, será necesario disminuir o incluso anular la velocidad de translación cuando haya que rotar un

ángulo considerable, ya que de lo contrario el espacio necesario para girar producirá posiblemente una invasión del espacio prohibido ocupado por el obstáculo situado frente al robot.

El sistema de navegación nos proporciona en todo momento una consigna de movimiento $\vec{F}$ que indica la posición espacial hacia la que debería moverse el robot. Plantearemos un conjunto de ecuaciones que permitirán obtener directamente el comando de velocidad a enviar al controlador. Tendremos en cuenta cuatro ideas centrales:

- Cuando la dirección deseada de movimiento difiera mucho de la dirección actual, la velocidad translacional deberá reducirse. Esta reducción será proporcional al error de dirección, es decir, a la diferencia entre la dirección actual y la deseada.
- Cuando se aproxime a un objetivo definido en términos geométricos, el robot deberá "atracar" en su punto de destino, a saber, deberá aminorar su velocidad translacional para evitar pasarse de largo. Esta reducción se efectuará linealmente y será proporcional a la distancia entre la posición estimada del robot y la posición del objetivo. Cuando el robot se aproxime a un subobjetivo que no sea el objetivo final no se realizará maniobra de atraque; simplemente, cuando esté suficientemente cerca, el subobjetivo actual se sustituirá por el siguiente. Es este caso el control no se ve alterado.
- La velocidad de rotación transmitida al controlador deberá ser mayor cuanto mayor sea la diferencia entre la dirección deseada de movimiento y la dirección actual. Mientras más parecidas sean estas direcciones menor será la velocidad de rotación aplicada. Si son iguales el robot se dirige hacia donde queremos: es ente caso la velocidad de rotación será cero.
- Las velocidades de rotación y translación tienen unas cotas máximas v_r y v_t definibles por el usuario.

Por último, un pequeño detalle a tener en cuenta es que la escala de unidades de los comandos del robot viene dada en decipulgadas (décimas de pulgada) para las posiciones y en decigrados (décimas de grado) para los ángulos, mientras que la escala de unidades del navegador está en pulgadas y radianes. Esto se tendrá en cuenta haciendo las oportunas conversiones.

Las ecuaciones 7.1, 7.2 y 7.3 describen cómo obtener las velocidades de translación y rotación en función de la dirección deseada de movimiento y de la posición angular del robot. Las funciones correspondientes pueden verse en la figura 7.4.

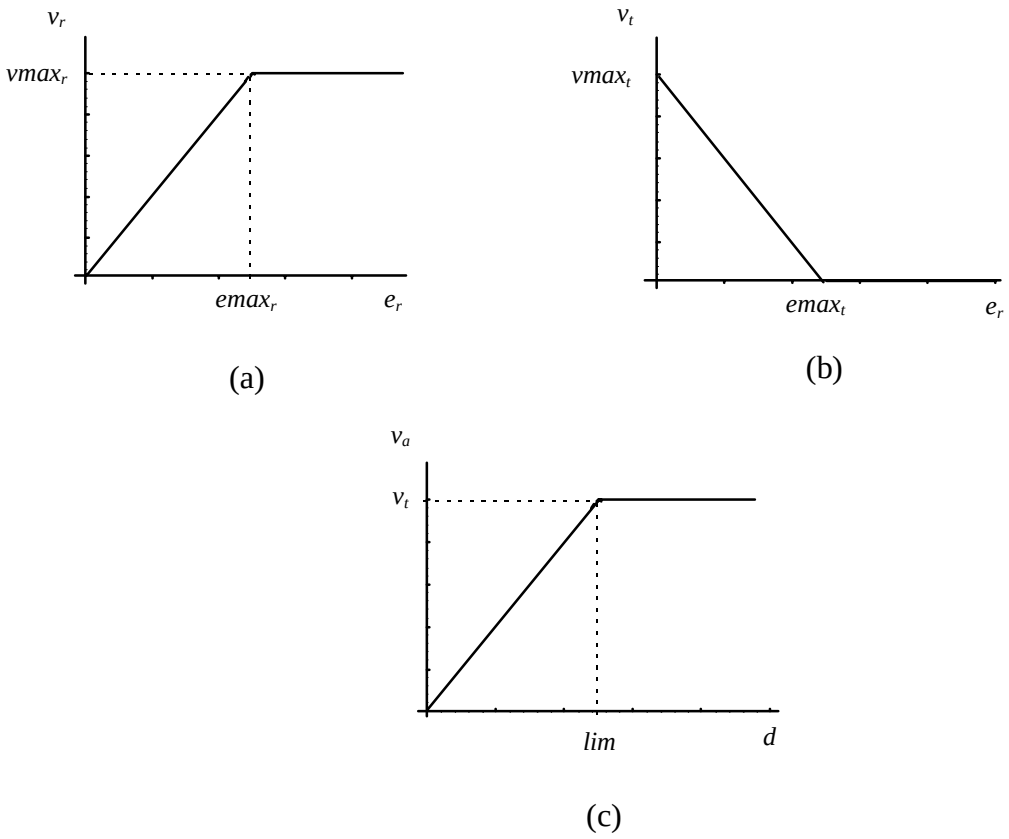


Figura 7.4. Funciones de cálculo de velocidad rotacional (a)*, translacional* (b) *y de atraque* (c)

$$v_r = \frac{vmax_r}{emax_r} e_r \Leftrightarrow |e_r| < emax_r$$
$$v_r = vmax_r \text{ en caso contrario} \qquad (7.1)$$

donde v_r es la velocidad rotacional, $vmax_r$ es la velocidad rotacional máxima, $emax_r$ es el error a partir del cual la velocidad de rotación será la máxima permitida y e_r es el error rotacional actual.

$$v_t = vmax_t \left(1 - \frac{|e_r|}{emax_t}\right) \Leftrightarrow |e_r| < emax_t$$
$$v_t = 0 \text{ en caso contrario} \qquad (7.2)$$

donde v_t es la velocidad translacional, $vmax_t$ es la velocidad translacional máxima, $emax_t$ es el error a partir del cual la velocidad de translación será cero y e_r es el error rotacional actual.

$$v_a = v_t \frac{d}{limite} \Leftrightarrow d < limite$$
$$v_a = v_t \quad \text{en caso contrario} \tag{7.3}$$

donde v_a es la velocidad translacional en condiciones de atraque, v_t es la velocidad translacional calculada con la ecuación 7.2, *d* es la distancia al punto de destino y *límite* es la distancia por debajo de la cual se aplica la reducción de atraque.

7.5 Reflexiones sobre el diseño y desarrollo de programas en el ámbito de la investigación

Llegados a este punto no queremos dejar pasar la oportunidad de reflexionar sobre las diferencias metodológicas existentes entre la programación para investigación y la programación para desarrollo. La dificultad básica de la programación para investigación es que las especificaciones del sistema a desarrollar cambian *permanentemente*, por lo que la necesidad de utilizar diseños robustos —hecho deseable en cualquier proyecto— es aquí aún más apremiante.

En un entorno en el que las especificaciones se modifican constantemente y a gran velocidad, la mayor parte de las normas básicas de diseño siguen imperturbables. No obstante, la alta tasa de modificaciones arroja una nueva luz sobre algunas normas comunes de diseño. En estas circunstancias es inevitable que la calidad del código producido no sea tan elevada como debiera ser en un proyecto de desarrollo. Lo deseable es, tras el proceso de investigación y una vez esclarecidas las técnicas útiles para la resolución del problema, rediseñar y reimplementar el sistema en la fase de desarrollo.

El lenguaje Lisp ha demostrado en numerosas ocasiones ser un firme candidato para tareas de prototipado rápido. Debido a las limitaciones de éste para ejecutarse en máquinas modestas —como es la que equipa al robot NOMAD-200— y a las necesidades de respuesta en tiempo real, hemos trabajado fundamentalmente en C++, lenguaje cuyas características, utilizadas con precaución, han producido un resultado suficientemente robusto en estas circunstancias

extremas. Para probar algunos algoritmos colaterales, como el de exploración en espacios de estados utilizando cargas ficticias generalizadas, hemos utilizado Lisp, precisamente por sus facilidades de prototipado.

Comentamos seguidamente alguna de las ideas surgidas a lo largo del proceso de programación de los métodos desarrollados en esta tesis; algunos de los comentarios son relativos a técnicas específicas situadas dentro del paradigma de la Programación Orientada a Objetos:

- **rediseñar lo mínimo**: no es razonable rediseñar el sistema cada vez que se efectúa alguna modificación. El desorden se va acumulando en el código de manera que periódicamente se hace necesaria una reestructuración. Es frecuente que algunas funciones implementadas y variables ya no se usen, o que métodos públicos puedan pasar a ser privados, etc. En estas reestructuraciones aparecen a veces variables redundantes o contradictorias, y nombres de variable y de métodos que no responden bien a su propósito actual, sino a aquél para el que fueron concebidos inicialmente.
- **memoria dinámica**: el manejo de memoria dinámica —imprescindible al trabajar con grafos— plantea habitualmente numerosos problemas. En este sentido Lisp es una buena alternativa frente a C++ por su excelente abstracción del trabajo con punteros, aunque su eficiencia es mucho más baja.
- **acceso a variables**: es norma generalizada la implementación de métodos de acceso a lectura y modificación de las variables de un objeto. Trabajando en investigación sin embargo no es recomendable esta práctica, porque las variables de clase cambian constantemente según se remodela la teoría, lo que obligaría a efectuar las modificaciones también en los métodos de acceso.
- **métodos virtuales**: hay que ser cuidadoso con el uso de métodos virtuales. Éstos constituyen una de las herramientas más potentes del paradigma de la programación orientada a objetos, pero el curso de ejecución de un programa que haga uso intensivo de ellos tiende a ser bastante complejo, pudiendo aparecer algunos problemas al introducir modificaciones en el código. Ningún procedimiento debe ser virtual *hasta que no se demuestre lo contrario*.

- **ocultación**: mantener un cierto rigor en la ocultación de variables y métodos es bastante complicado. Elementos que inicialmente se concibieron como ocultos tienen eventualmente —al ser modificado un algoritmo— que ser consultados desde otra clase, de modo que deben pasar de privados a públicos, y viceversa, elementos que se concibieron como públicos se observa posteriormente que por acumulación de modificaciones han dejado de ser consultados desde el exterior de la clase, pudiendo pasar de públicos a privados (aunque esto es ante todo una decisión de diseño).

7.6 Resultados

En este capítulo hemos abordado algunos aspectos relevantes referidos al diseño y la implementación *software*. En el apartado de diseño, hemos expuesto la estructura de clases que soporta la implementación tanto del simulador como del programa que controla al robot NOMAD-200. Es de resaltar la configuración *en capas* del código, de manera que es posible reutilizarlas independientemente. Del mismo modo, aprovechando las características de la programación orientada a objetos (en particular la definición de clases abstractas), es posible adaptar las clases a diferentes plataformas. En esta línea, tanto el simulador como el programa del robot utilizan las mismas clases básicas, particularizadas convenientemente (piénsese, a modo de ilustración, las diferencias entre obtener la lectura de un sensor real y obtener la lectura de un sensor simulado).

En el apartado de implementación, la contribución más importante es la del módulo de control para el robot NOMAD. Dicho módulo establece un interfaz entre las instrucciones de movilidad proporcionadas por el sistema de navegación y los comandos de velocidad que acepta la plataforma, y se basa en el conjunto de ecuaciones de control desarrolladas en este capítulo.

8. Resultados experimentales

En este capítulo mostraremos una serie de resultados empíricos obtenidos utilizando el software desarrollado sobre el robot NOMAD-200. Estos resultados hacen referencia al comportamiento de los campos de potencial con cargas ficticias, a la detección de LSRs y a la construcción y uso de modelos cognitivos. Todos los ejemplos presentados en este capítulo han sido obtenidos de operaciones reales del robot en el Laboratorio de Visión Artificial y en los pasillos de la segunda planta de la Facultad de Informática de la UPM, siempre con objetos reales (mesas, terminales, estanterías) y en algunos casos con personas moviéndose dentro del campo de acción del robot.

8.1 Navegación con el método del potencial y cargas ficticias

Como se explicó en el capítulo 2, el método del potencial con cargas ficticias permite definir un sistema de navegación que puede seguir un rumbo determinado o intentar alcanzar unas coordenadas de destino, capaz de eludir obstáculos de pequeña magnitud tanto estáticos como móviles. En el entorno del robot, estos obstáculos de pequeña magnitud están constituidos fundamentalmente por el mobiliario (mesas, armarios, bancos) y las personas próximas a la plataforma.

Recordamos que, cuando el sistema detecta un mínimo local, sitúa una carga ficticia en la posición determinada por la ecuación 2.3 y continúa la misión.

En la figura 8.1 mostramos la trayectoria y las lecturas de los sensores en una misión sencilla en la que se envía al robot a un punto situado tras un armario. El navegador consigue eludir el armario y llegar al punto de destino (*goal*). Seguidamente se envía el robot de nuevo al punto de partida (*start*). En este caso se detectan varios mínimos locales (representados en la figura mediante cruces); utilizando cargas ficticias, el navegador es capaz de regresar al punto de partida. La figura sirve también como ejemplo del módulo de pilotaje, ya que la trayectoria representada es la trayectoria real seguida por el robot.

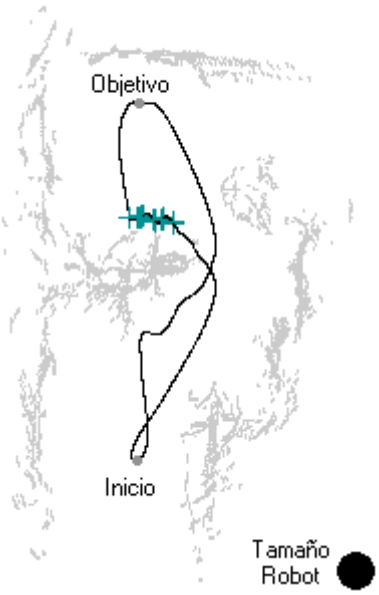


Figura 8.1. Ejemplo de navegación con potencial artificial y cargas ficticias

8.2 Detección de Lugares Sensorialmente Relevantes basada en el gradiente

El alcance de los sensores de ultrasonidos de la plataforma NOMAD-200 es de aproximadamente 255 pulgadas, es decir, unos 6,5 metros, mientras que el alcance de los sensores del simulador se fijó, por motivos de eficiencia, en 70 pulgadas (unos 1,8 metros). Estas diferencias de alcance, junto a los problemas introducidos por los sensores reales (ruido, imprecisiones, etc.) nos obligan a reconsiderar aquí los umbrales de detección de LSRs.

En la figura 8.2 se muestra un histograma de distribución de módulos de gradiente sensorial tomados de una misión real (figura 8.3) con 28788 muestras. Como puede observarse, la distribución es similar a la de los datos obtenidos con el simulador: aproximadamente una curva hiperbólica, con la mayor parte de los casos concentrados en valores bajos. Esto es consistente con nuestra propuesta, ya que el número de puntos de detección de LSRs puede hacerse tan bajo como queramos sin más que incrementar el umbral de detección. Aplicando la ecuación 3.4, y teniendo en cuenta que el alcance mínimo de los sensores reales es de unas 6 pulgadas, podemos determinar que el umbral óptimo se encuentra en el intervalo (37.35, 49.8).

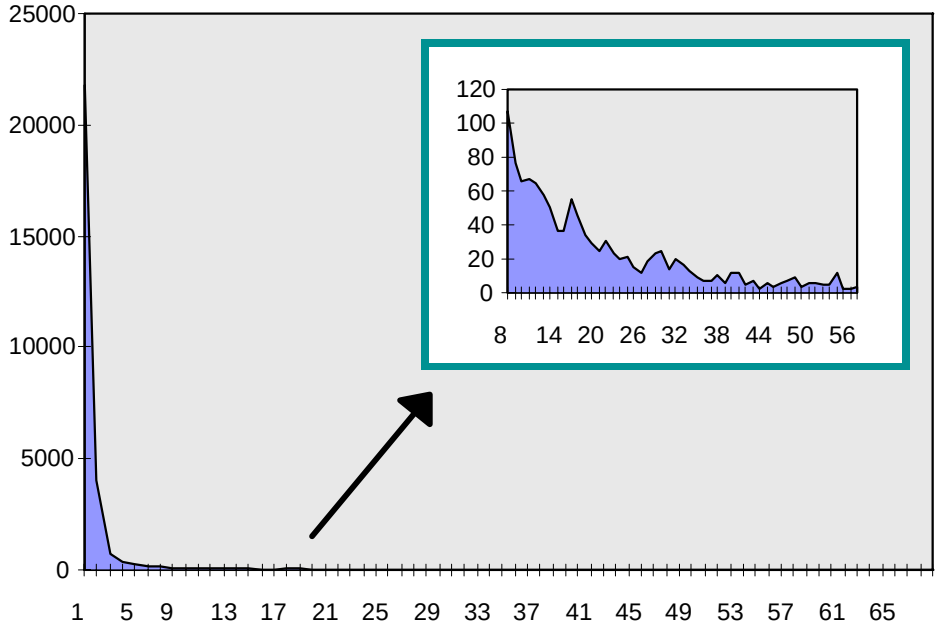


Figura 8.2. Histograma de distribución de módulos de gradiente sensorial en un caso real

Aplicando estos valores a los datos reales podemos ver que nos movemos entre el 0,2% y el 0,5% de las muestras. Si tenemos en cuenta que en las 28788 iteraciones el robot recorrió 5299 pulgadas, podemos concluir que, en promedio, en cada iteración se recorren 0,184 pulgadas. Con ambos datos podemos obtener cada cuántas pulgadas (siempre en promedio) detectaremos un LSR, utilizando la ecuación 8.1.

$$d_{LSR} = \frac{0.184}{p} \qquad (8.1)$$

donde d_{LSR} es la distancia promedio entre LSRs y p es el porcentaje (en tanto por uno) de LSRs detectados.

Aplicando esta ecuación obtenemos que la distancia promedio entre LSRs se encuentra entre 36,8 pulgadas (0,93m) y 92 pulgadas (2,34m). Naturalmente todos estos valores son sólo una referencia burda puesto que el número de LSRs detectados dependerá en todo caso de las condiciones del entorno.

Hemos de recordar, además, que el hecho de que un LSR sea detectado no quiere decir que sea añadido al modelo: esto depende de que en las proximidades exista o no un LSR detectado

previamente. Si existe un LSR previo, éste se modificará para incorporar los nuevos datos sobre medidas sensoriales y cuadrantes (ver capítulos 3 y 4).

8.3 Errores en la estimación odométrica y goniométrica

Hasta este punto hemos supuesto que la información odométrica (localización cartesiana del robot) y goniométrica (posición angular) eran relativamente exactas, pero esto en realidad no es así. Los mecanismos incorporados en la plataforma para determinar su posición y orientación acumulan grandes errores a medida que ésta se mueve. Para garantizar cierta fiabilidad en la información de la localización del robot, imprescindible durante la construcción y uso de mapas con información sensorial pobre[12], es necesario utilizar mecanismos de corrección de los desfases sistemáticos en el posicionamiento de la plataforma.

En la figura 8.3 se muestra una misión sin corrección odo-goniométrica, donde se observa que el sistema falla en el regreso a su posición de origen debido al error acumulado.

Afortunadamente, el error de estimación es sistemático: siempre se acumula en la misma dirección. Esto permite afinar la precisión del sistema del propio robot incorporando una corrección *software* que, en cada paso de iteración del navegador, modifique la posición proporcionada por el robot añadiéndole el error acumulado estimado. Los procedimientos para estimar este error y para modificar la posición proporcionada por el robot se exponen en el apartado 8.3.1.

Por desgracia, aún utilizando mecanismos de corrección, siempre se acumulará algún error, y, por pequeño que éste sea, al cabo de muchas iteraciones el desfase entre la posición teórica y la real terminará alcanzando magnitudes apreciables. Para dar cuenta de este problema

[12] Utilizando un sistema sensorial rico, como por ejemplo la visión artificial, cada lugar del mundo podría identificarse unívocamente, y de este modo sería posible prescindir de los sistemas de posicionamiento, puesto que la posición estaría completamente definida por la identificación sensorial. En nuestro robot, que sólo dispone de sensores de ultrasonidos, esto no es posible, ya que pueden existir muchos lugares del entorno con información sensorial similar.

hemos definido un mecanismo adicional, basado en las ideas propuestas por Kurz [Kurz, 1993; 1996] aplicadas a nuestros LSRs. Este método se expone en el punto 8.3.2.

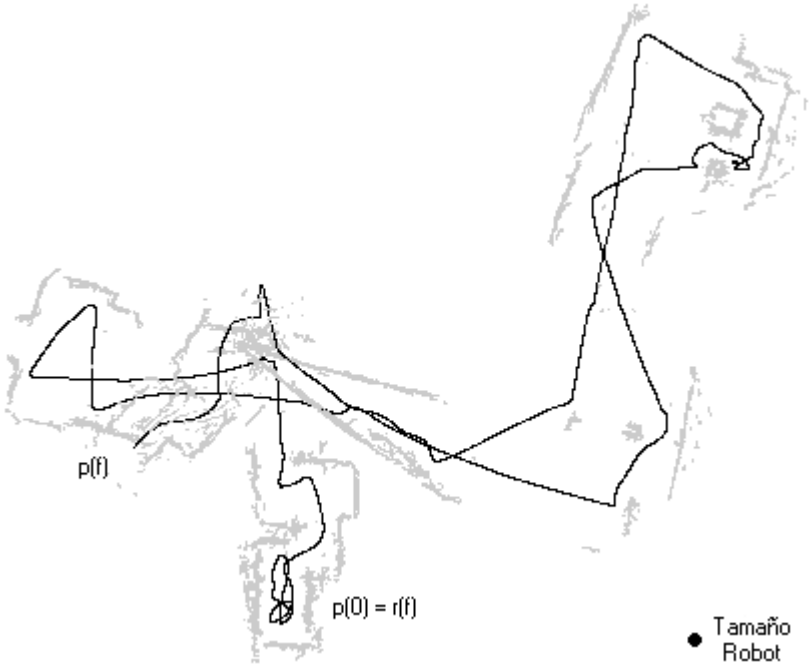


Figura 8.3. Misión de exploración con errores goniométrico y odométrico sin corregir. p(0) representa la posición odométrica inicial, p(f) la posición odométrica final y r(f) la posición real final.

8.3.1 Calibrado y corrección de los errores sistemáticos

En esta sección intentaremos formalizar el error final de posicionamiento cometido en N iteraciones con respecto a los distintos tipos de error cometidos en cada iteración. Esto nos permitirá (a) estimar los parámetros del error a partir del error final de posicionamiento tomado de datos experimentales, y (b) corregir las estimaciones de posicionamiento proporcionadas por el robot aplicando los parámetros estimados.

En la figura 8.4 se presenta un esquema del problema de posicionamiento relativo a la iteración i, donde $\vec{o}_i$ se refiere a la posición obtenida del robot por realimentación odométrica, $\vec{r}_i$ se refiere a la posición real (desconocida) del robot, $\alpha_o(i)$ es el ángulo de rotación de la base del robot obtenido mediante la realimentación goniométrica y $\alpha_r(i)$ es el ángulo real (desconocido) de la base del robot. El vector $\vec{e}_i$ representa el error odométrico total cometido.

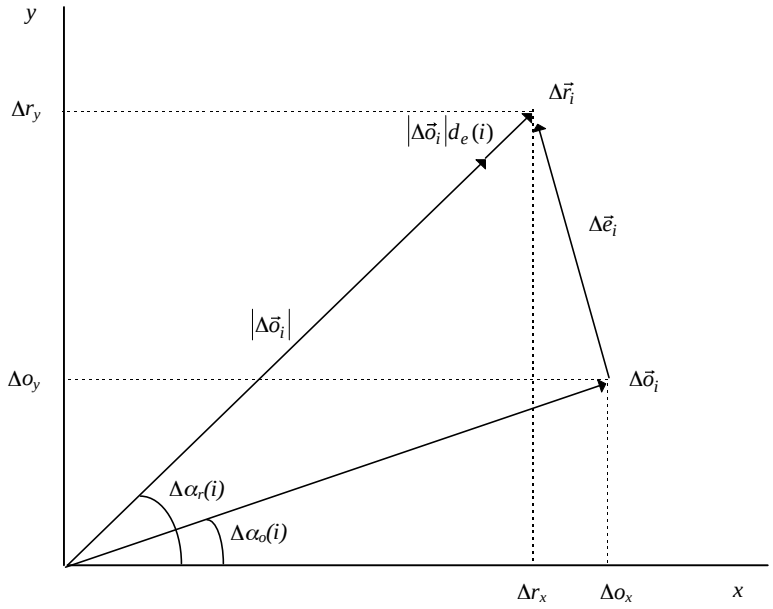


Figura 8.4. Esquema de los errores de posicionamiento acumulados en una iteración

Como demostraremos, la ecuación 8.8 determina el incremento del error en la iteración *i*, y la ecuación 8.9 el error total acumulado tras una misión. En efecto, definido el incremento del error como

$$\Delta\vec{e}_i = \Delta\vec{r}_i - \Delta\vec{o}_i \tag{8.2}$$

podemos expresar sus componentes a través de las proyecciones en *x* e *y*:

$$\begin{aligned} \Delta e_x(i) &= |\Delta\vec{r}_i| \cos\left(\Delta\alpha_r(i)\right) - |\Delta\vec{o}_i| \cos\left(\Delta\alpha_o(i)\right) \\ \Delta e_y(i) &= |\Delta\vec{r}_i| \operatorname{sen}\left(\Delta\alpha_r(i)\right) - |\Delta\vec{o}_i| \operatorname{sen}\left(\Delta\alpha_o(i)\right) \end{aligned} \tag{8.3}$$

Es posible relacionar los incrementos reales con los teóricos introduciendo los parámetros $\alpha_e(i)$ (factor del error angular debido a rotaciones), $\alpha_t(i)$ (factor del error angular debido a translaciones) y $d_e(i)$ (factor del error translacional debido a translaciones). Hemos comprobado experimentalmente que los errores translacionales debidos a rotaciones son despreciables. Las relaciones entre los incrementos de posición teóricos y los reales se muestran en las ecuaciones 8.4 y 8.5.

$$|\Delta\vec{r}_i| = |\Delta\vec{o}_i|\left(1 + d_e(i)\right) \tag{8.4}$$

$$\Delta\alpha_r(i) = \Delta\alpha_o(i)\alpha_e(i) + \left|\Delta\vec{o}_i\right|\left(1 + d_e(i)\right)\alpha_t(i) \qquad (8.5)$$

Supondremos que $\alpha_e(i)$, $\alpha_t(i)$ y $d_e(i)$ son constantes a lo largo de las iteraciones, con la salvedad —provocada por una realidad observacional— de que tendremos dos valores para $\alpha_e(i)$, dependiendo de si $\Delta\alpha_o(i)$ es positivo o negativo. A estos dos posibles valores los denominaremos α^+_e y α^-_e. Esta suposición es una simplificación, ya que en la realidad los errores cometidos están sujetos a factores locales variables con el tiempo, como son el estado del piso, la aceleración aplicada a los motores, etc. Una estimación fina del error calculada sin efectuar esta simplificación sería en sí misma objeto de otra tesis.

Definiendo el ángulo corregido acumulado como

$$\begin{aligned} \alpha_{acum}(i) &= \alpha_{acum}(i-1) + \Delta\alpha_o(i)\alpha_e + \left|\Delta\vec{o}_i\right|(1 + d_e)\alpha_t \\ &\text{con} \quad \alpha_e = \alpha_e^+ \Leftrightarrow \Delta\alpha_o(i) \geq 0 \\ &\text{y} \quad \alpha_e = \alpha_e^- \quad \text{en caso contrario} \end{aligned} \qquad (8.6)$$

y sustituyendo en 8.3, podemos concluir que

$$\begin{aligned} \Delta e_x(i) &= \left|\Delta\vec{o}_i\right|(1 + d_e)\cos\left(\alpha_{acum}(i)\right) - \left|\Delta\vec{o}_i\right|\cos\left(\Delta\alpha_o(i)\right) \\ \Delta e_y(i) &= \left|\Delta\vec{o}_i\right|(1 + d_e)\,\mathrm{sen}\left(\alpha_{acum}(i)\right) - \left|\Delta\vec{o}_i\right|\,\mathrm{sen}\left(\Delta\alpha_o(i)\right) \end{aligned} \qquad (8.7)$$

y utilizando la posición odométrica en *i-1*,

$$\begin{aligned} \Delta e_x(i) &= \left|\Delta\vec{o}_i\right|(1 + d_e)\cos\left(\alpha_{acum}(i)\right) - \left(\vec{o}_x(i) - \vec{o}_x(i-1)\right) \\ \Delta e_y(i) &= \left|\Delta\vec{o}_i\right|(1 + d_e)\,\mathrm{sen}\left(\alpha_{acum}(i)\right) - \left(\vec{o}_y(i) - \vec{o}_y(i-1)\right) \end{aligned} \qquad (8.8)$$

Al cabo de *N* iteraciones, el error acumulado será igual a

$$\begin{aligned} e_x &= \sum_{i=0}^{N-1} e_x(i) \\ e_y &= \sum_{i=0}^{N-1} e_y(i) \end{aligned} \qquad (8.9)$$

Para obtener los valores de los parámetros del error, α_e^+, α_e^-, α_t y d_e, estimaremos sus valores de modo que el error predicho coincida con el error real obtenido tras una serie de misiones.

La metodología de ajuste es la siguiente:

a) Partiendo de $\alpha(0) = 0$, realizar una misión de exploración relativamente larga, registrando en cada iteración los valores sin corrección de $\alpha_o(i)$ y de $\vec{o}_i$.

b) Medir el error angular final acumulado sobre la base del robot (α_{final}), y calcular el ángulo total girado durante la misión en la dirección positiva (α_{acum}^+) y en la dirección negativa (α_{acum}^-), así como el desplazamiento total d_{acum}.

c) Obteniendo estos datos de k misiones podemos realizar, sobre la ecuación 8.10, un ajuste por mínimos cuadrados de los parámetros. Finalmente, con la ecuación 8.9 podemos ajustar d_e.

$$\alpha_{acum}^+(k)\alpha_e^+ + \alpha_{acum}^-(k)\alpha_e^- + d_{acum}(k)\alpha_t = \alpha_{final}(k) \qquad (8.10)$$

Una vez determinados los parámetros del error, disponemos de un procedimiento para estimar el error acumulado para cada iteración i utilizando la ecuación 8.8. Conociendo el error en la iteración *i-1*, podemos calcular el error en i con la ecuación 8.11.

$$\begin{aligned} e_{acum,x}(i) &= e_{acum,x}(i-1) + e_x(i) \\ e_{acum,y}(i) &= e_{acum,y}(i-1) + e_y(i) \end{aligned} \qquad (8.11)$$

En la tabla 8.1 se muestran los datos tomados de una serie de misiones realizadas con el robot NOMAD-200. Utilizando estos datos hemos obtenido los valores para los parámetros del error, que son: α_e^+ = 0.00153697 radianes de error / radian rotado en el sentido positivo, α_e^- = -0.00198783 radianes de error / radian rotado en el sentido negativo y α_t = 0.000152126 radianes de error / pulgada recorrida. El parámetro d_e apenas afecta al comportamiento del sistema, por lo que hemos tomado d_e = 0 pulgadas de error / pulgada recorrida.

Utilizando la corrección odo-goniométrica de la ecuación 8.11 hemos modificado los resultados de la misión de la figura 8.3. Estos resultados se muestran en la figura 8.5.

α^{+}_{acum}	α^{-}_{acum}	d_{acum}	α_{final} (error medido)
119,38	0	0	0,196
0	106,81	0	-0,196
29,64	29,39	2834,90	0,349
50,99	37,00	4668,16	0,698
42,34	43,64	1616,97	0,122
57,49	47,55	4771,41	0,785
36,92	43,81	2436,59	0,393

Tabla 8.1. Datos recabados en varias misiones

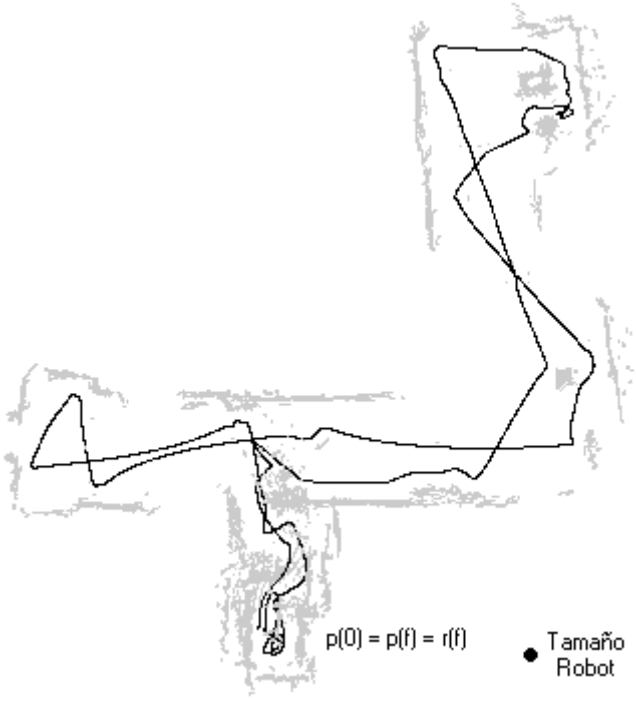


Figura 8.5. La misma misión de la figura 8.3 con corrección odo-goniométrica simulada. p(0) representa la posición odométrica inicial, p(f) la posición odométrica final y r(f) la posición real final.

8.3.2 Corrección cartesiana *basada en LSRs*

La corrección basada en LSRs se fundamenta en el reconocimiento de un lugar visitado previamente (el LSR) y la consiguiente modificación de la posición estimada del robot para aproximarla a la posición almacenada en el lugar.

Mientras mayores sean las capacidades de reconocimiento del sistema, menor será la probabilidad de confundir un lugar con otro. Téngase en cuenta que la capacidad de reconocimiento depende de dos factores: (a) la naturaleza de los sistemas sensoriales utilizados y (b) la calidad del proceso de reconocimiento.

En lo que respecta al punto (a), y como ya anticipábamos, la información suministrada por los sensores de ultrasonidos no es suficientemente rica para permitir la identificación unívoca de lugares. Para mejorar esta información disponemos de otro tipo de sensores: la propia odometría del robot.

En lo que respecta al punto (b), utilizando los datos aproximados sobre la posición del robot, junto con las lecturas de los ultrasonidos, podemos construir un reconocedor basado en la distancia euclídea (como ya se expuso en el capítulo 4) que identifique, en los puntos de alto gradiente, el LSR asociado a dicho punto. Recuperando las coordenadas del LSR identificado podemos actualizar la posición estimada del robot con arreglo a la ecuación 8.12, que sustituye a la ecuación 8.10 en los casos en los que se esté detectando LSR. En caso contrario $e_{acumLSR,x}(i) = e_{acum,x}(i)$ y $e_{acumLSR,y}(i) = e_{acum,y}(i)$.

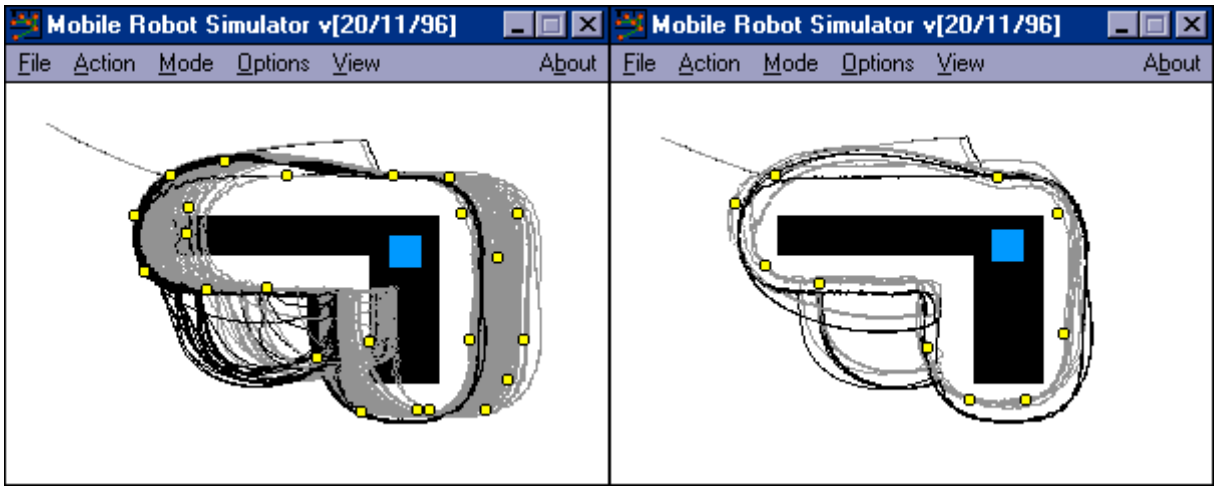
$$\begin{aligned} e_{acumLSR,x}(i) &= e_{acum,x}(i) + \lambda\Big(LSR_x(i) - \big[o_x(i) + e_{acum,x}(i)\big]\Big) \\ e_{acumLSR,y}(i) &= e_{acum,y}(i) + \lambda\Big(LSR_y(i) - \big[o_y(i) + e_{acum,y}(i)\big]\Big) \end{aligned} \qquad (8.12)$$

donde $(LSR_x(i), LSR_y(i))$ se refiere a las coordenadas del LSR, y λ es un parámetro que regula la magnitud del salto de la corrección.

El propósito del parámetro λ es que las modificaciones no sean bruscas, porque de lo contrario, si la detección de LSR comete un error ocasional (debido, por ejemplo a modificaciones en el entorno), el sistema se perdería. Con este parámetro, sin embargo, un error ocasional no afectará al rendimiento del método.

En la figura 8.6 (a y b) podemos ver un ejemplo, generado en el simulador, del mecanismo de compensación odométrica basado en LSRs, comparado con el resultado obtenido sin compensación. En el ejemplo se fijó una meta inalcanzable para el robot (la meta se encuentra dentro de un objeto) para que éste rotase indefinidamente alrededor de su punto de destino.

Las líneas grises indican la trayectoria teórica seguida por el robot; las líneas en negro, la trayectoria real. Como puede observarse, el error crece indefinidamente cuando no hay corrección, permaneciendo acotado cuando sí la hay. En la figura 8.6a se comprueba también que el error acumulado origina la creación de LSRs inexistentes debido a que éste impide la identificación de algunos LSRs que ya están en el modelo. Al no ser identificados, el sistema crea nuevos nodos.



(a) (b)

Figura 8.6. Errores odométricos acumulados con d_e=0.02; (a) sin corrección, (b) acotado indefinidamente gracias a la corrección basada en LSRs

8.3.3 Corrección angular *basada en LSRs*

La compensación expuesta en el apartado anterior corrige solamente errores odométricos. Para los errores goniométricos hemos desarrollado otro tipo de compensación basada en los LSRs. La idea consiste en que, una vez identificado un LSR, podemos rotar la torreta realizando un descenso del gradiente de la distancia entre la imagen sensorial actual y la imagen sensorial almacenada en el LSR. Para calcular esta distancia puede usarse la ecuación 4.5, propuesta en el capítulo 4 para la identificación de nodos. Si rotamos en la dirección adecuada, dicha distancia irá disminuyendo hasta hacerse mínima en el momento en que la coincidencia entre las imágenes sensoriales se hace máxima. En estas condiciones, podemos afirmar que el ángulo de rotación de la torreta (en sentido contrario) con respecto a la base del robot corresponde al desfase angular acumulado desde la última vez que se visitó el LSR (ecuación 8.13).

$$\alpha_{acum}(i) = -\tau(i) \qquad (8.13)$$

siendo τ(i) la rotación de la torreta obtenida del robot en la iteración *i*.

El algoritmo de recalibración angular en LSR es el siguiente:

RECALIBRACIÓN-ANGULAR (LSR, INCREMENTO)

1) Medir D-ACTUAL utilizando la ecuación 4.5.
2) Rotar INCREMENTO en el sentido positivo. Medir D-POS. Volver a la posición inicial.
3) Rotar INCREMENTO en el sentido negativo. Medir D-NEG. Volver a la posición inicial.
4) Si (D-ACTUAL < D-POS) && (D-ACTUAL < D-NEG) terminar; no hace falta recalibración.
5) Si (D-POS < D-NEG) DIR=1 de lo contrario DIR=-1.
6) D-ANTERIOR = D-ACTUAL; TOTAL=0.
7) Mientras D-ACTUAL <= D-ANTERIOR, hacer
   a) Rotar DIR * INCREMENTO; TOTAL = TOTAL + INCREMENTO.
   b) Hacer D-ANTERIOR = D-ACTUAL.
   c) Medir D-ACTUAL en la nueva posición.
8) En este punto TOTAL almacena el ángulo total rotado. El error goniométrico acumulado en la base es -β, siendo β la posición angular actual de la torreta.

Tras la ejecución de este algoritmo, la torreta queda alineada con el sistema de referencia del LSR identificado.

Este procedimiento presenta el inconveniente de que es necesario detener el proceso en curso y recalibrar en cada LSR identificado. Una opción alternativa podría recalibrar cada cierto número de LSRs identificados, o cuando la identificación se considerase muy segura.

8.3.4 Desfase de los sensores

Una última consideración sobre el error acumulado es que el desfase angular debe tenerse en cuenta para corregir la posición real de los sensores. Siendo $\theta_s(0)$ la orientación inicial del sensor *s*, podemos calcular el error de orientación acumulado en la iteración *i* mediante la ecuación 8.7 o bien utilizando el algoritmo de recalibración angular, y con este error obtener la orientación real del sensor (ecuación 8.14)

$$\theta_s(i) = \theta_s(0) + \tau(i) + \alpha_{acum}(i) \qquad (8.14)$$

siendo $\tau(i)$ la rotación de la torreta obtenida del robot en la iteración i.

El desfase de los sensores puede ignorarse a efectos del cálculo de la fuerza repulsiva si también se ignora al aplicar dicha fuerza al robot, puesto en este caso los errores se cancelan. Es por esto que en los enfoques puramente reactivos se ignoran habitualmente los errores cometidos tanto en translación como en rotación.

Sin embargo, en el momento en que introducimos una *intención*[13] de rumbo en el sistema de navegación, necesitamos corregir los desfases angulares para asegurarnos de que el rumbo seguido es aproximadamente el deseado. Por otro lado, a la hora de modelar el mundo, es importante registrar correctamente la dirección real de medida de los sensores.

Como conclusión, al renunciar a un enfoque exclusivamente reactivo se hace necesario introducir corrección angular en la base del robot, y esta corrección obliga a su vez a introducir una corrección angular en la posición de los sensores.

8.3.5 Consideraciones sobre los diferentes tipos de corrección

La corrección basada en LSRs proporciona un método robusto para mejorar el conocimiento sobre la posición del robot, pero sólo funciona si el reconocimiento de LSRs trabaja adecuadamente. Esto quiere decir que, puesto que utilizamos la propia información odométrica para enriquecer las medidas de los sensores de ultrasonidos, dicha información no puede acumular un error excesivo. Por otro lado, cuando realizamos misiones de exploración no existen LSRs con los que recalibrar, de modo que la información inicial sobre la posición de los LSRs está también afectada por los errores odométricos.

La consecuencia de estos hechos es que la recalibración previa de los errores sistemáticos es fundamental si el entorno de trabajo es relativamente grande, y conveniente en cualquier

[13] Una de las ideas permanentemente presentes en esta tesis es que es justamente el concepto de *intención* el que establece diferencias irreconcialiables entre los enfoques reactivos puros y los enfoques híbridos.

caso porque ayuda a mejorar el comportamiento de la recalibración basada en LSRs. En nuestro sistema utilizamos ambos métodos.

En cuanto al problema de la incertidumbre en la información inicial obtenida durante la exploración, nos remitimos a la argumentación de Kurz [Kurz, 1996, pag. 239-240]: "*El método de estimación de la posición funciona bien si el mapa existe previamente […] Sin embargo, ¿es posible generar el mapa y usarlo al mismo tiempo para estimar la posición? Este es el problema general de aprender mapas con un robot móvil equipado con goniómetros para medir su posición: la estimación de la posición debe corregirse utilizando un mapa que sólo puede aprenderse usando dicha posición corregida. Los experimentos con el robot muestran que este problema puede resolverse. El filtro extendido de Kalman estima la posición con suficiente exactitud como para generar mapas estables siempre que se pueda garantizar una identificación correcta. La interdependencia entre la estimación de la posición y la construcción del mapa no conduce a un incremento de la incertidumbre.*"; hemos comprobado la validez de esta argumentación a través de simulaciones de mapas cognitivos construidos con error: estos mapas no "casan" con el mundo simulado, y sin embargo al ser utilizados producen resultados correctos porque, aunque la posición física sea diferente a la teórica, la posición teórica del robot sí que "casa" con el modelo.

En la figura 8.7 podemos ver el resultado de una misión real de exploración utilizando los sistemas de corrección. En negro se muestra el GLSR construido; en rojo, la trayectoria recorrida.

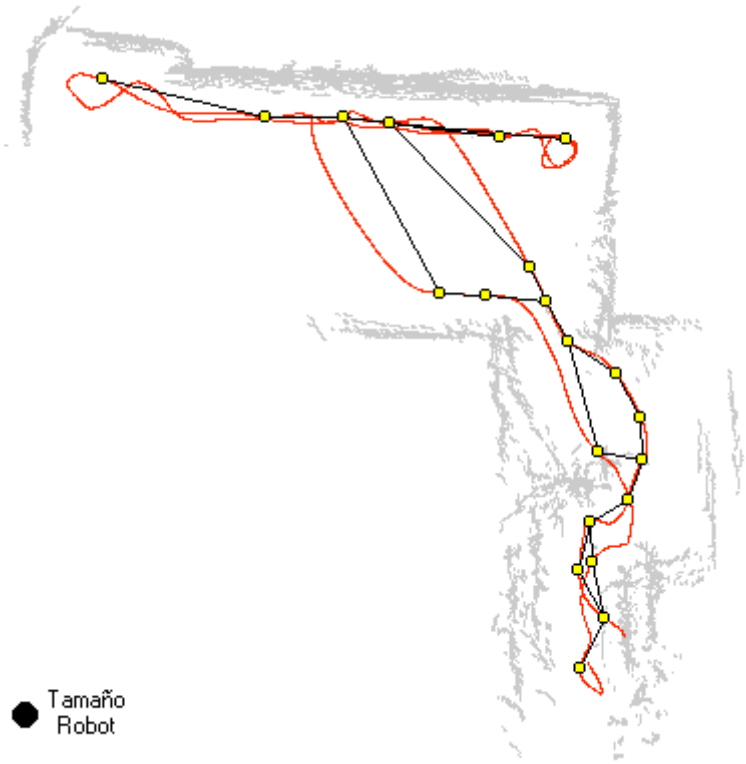


Figura 8.7. Resultado de una misión de exploración con corrección basada en LSRs

8.4 Resultados

Para finalizar este capítulo de pruebas experimentales sólo nos resta mostrar algunos resultados de misiones de construcción del modelo cognitivo y de uso posterior de los modelos construidos.

La construcción del modelo es un proceso tedioso debido a que el algoritmo ensaya exhaustivamente todos los cuadrantes inexplorados de cada LSR intentando determinar si existen conexiones útiles con algún otro LSR. El resultado, como ya es bien conocido, es un grafo de lugares sensorialmente relevantes (el GLSR) que puede posteriormente ser usado en nuevas misiones a través de alguno de los mecanismos de exploración expuestos en el capítulo 5.

En la figura 8.8 se muestra el modelo construido tras una misión de exploración, y en la figura 8.9 puede verse la trayectoria seguida por el robot en una misión utilizando el modelo

del la figura 8.8 y la planificación en oleadas. Las figuras 8.9 y 8.10 muestran el plan y la trayectoria recorrida durante una misión en un entorno más amplio.

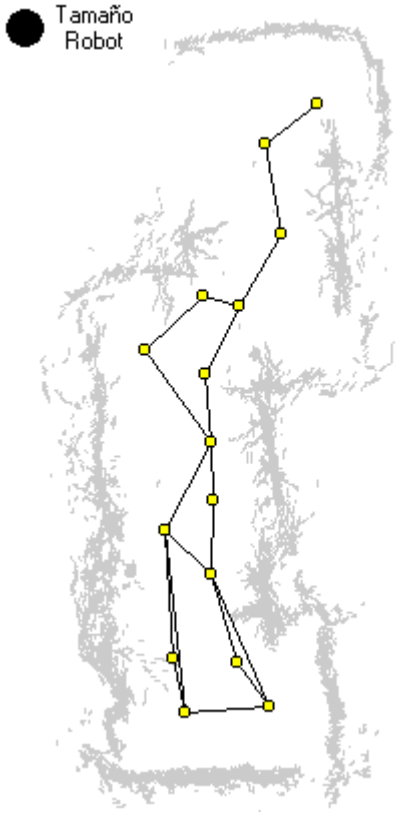


Figura 8.8. Modelo construido tras una misión de exploración

Figura 8.9. Trayectoria seguida por el robot mediante una planificación en oleadas con el modelo de la figura 8.8

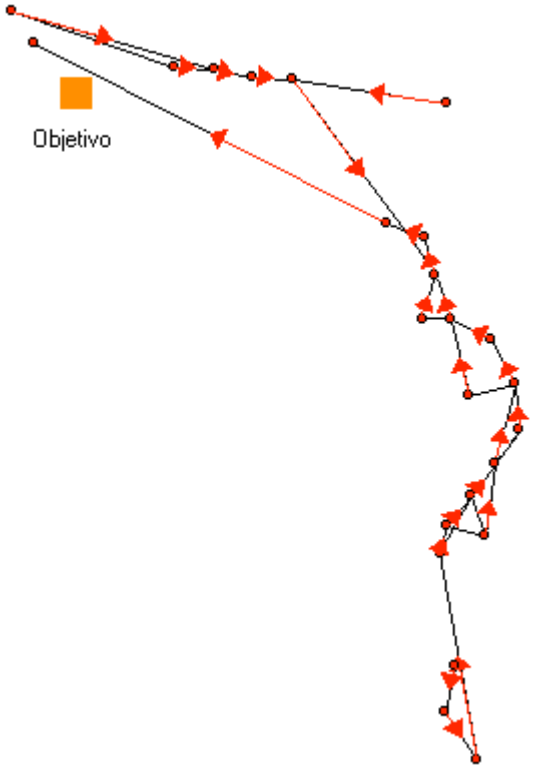


Figura 8.9. Sugerencias establecidas por la planificación en oleadas (en rojo) sobre un modelo previo (en negro)

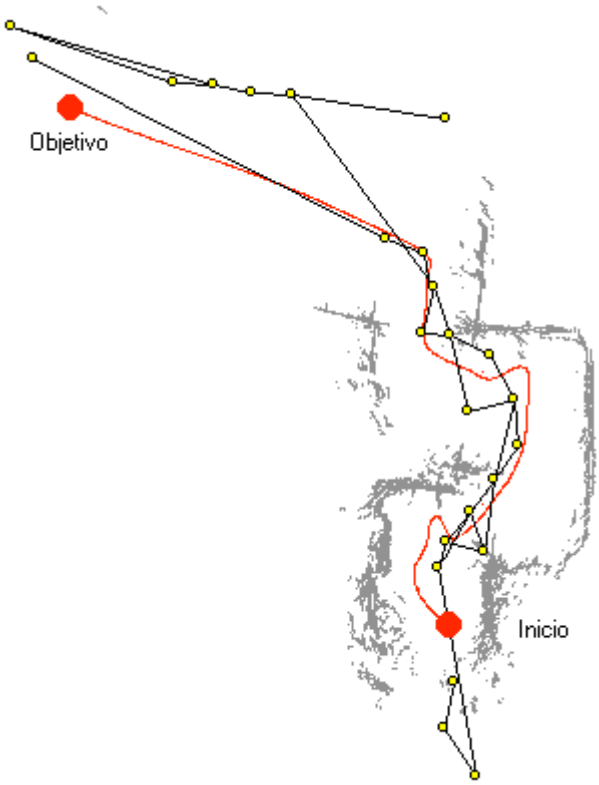


Figura 8.10. Trayectoria recorrida por el navegador utilizando las sugerencias de la figura 8.9.

9. Conclusiones

El resultado tangible de esta tesis es un sistema de navegación, aprendizaje, planificación y control para un robot móvil NOMAD-200 equipado con sensores de ultrasonidos, por un lado, y un simulador del robot, por otro. Este resultado se desgrana en una serie de contribuciones que podríamos polarizar en torno a varios temas, que comentaremos por separado. En orden de relevancia —a nuestro juicio— de las contribuciones, estos temas son:

- modelado autónomo del entorno basado en el gradiente sensorial
- elaboración de planes expresados mediante sugerencias
- navegación reactiva utilizando campos de potencial artificial con cargas ficticias
- otras aportaciones

9.1.1 Modelado autónomo

En cuanto al modelado autónomo, la aportación más importante es la del concepto de *gradiente sensorial* como herramienta de detección de lugares de interés (los *lugares sensorialmente relevantes*). Frente a los trabajos anteriores [Mataric, 1992; Walker *et al*, 1993; Kurz, 1996], que categorizan todos y cada uno de los registros sensoriales del robot, en nuestro sistema una posición sólo se intenta identificar cuando el gradiente sensorial lo indica, lo que supone evitarse alrededor del 95% de las comprobaciones. Por otro lado, la propia naturaleza del gradiente obliga a que los LSRs se sitúen en puntos estratégicos, tales como puertas, esquinas, etc. de modo que la estructura topológica construida con ellos permite establecer rutas casi óptimas con una carga computacional mínima. En este sentido nuestras

propuestas mejoran el interesante trabajo de Pierce y Kuipers [Pierce & Kuipers, 1994], ya que los nodos de los grafos que ellos construyen se sitúan en los rincones, lo que, aunque presenta otras ventajas, hace que las trayectorias elaboradas por su sistema disten de ser óptimas.

Para la detección de LSRs hemos estudiado dos posibilidades: la detección basada en el *módulo del gradiente sensorial* y la detección *utilizando mapas autoorganizados*. Ambas han ofrecido resultados satisfactorios, aunque la primera, en el caso de robots con simetría sensorial (como es el NOMAD-200), aprovecha mejor esta circunstancia.

Dentro de este mismo apartado debemos resaltar el algoritmo de *exploración basada en cuadrantes*, que constituye un procedimiento sistemático de generación de un *grafo de lugares sensorialmente relevantes*, estableciendo una relación de adyacencia entre los diferentes LSRs detectados en el entorno del robot.

9.1.2 Obtención de planes

En lo referente a la obtención de planes, esta tesis aporta un nuevo método de exploración en espacios de estado —la *búsqueda en oleadas*— que permite establecer planes entendidos como sugerencias sobre el GLSR. El resultado de este algoritmo es una sugerencia para cada LSR que indica la dirección que conduce al mejor sucesor con objeto de alcanzar el objetivo por el recorrido más corto.

Las ventajas de nuestro enfoque frente al trabajo de Payton, que establece *planes internalizados* sobre *grids* [Payton *et al*, 1990; Payton, 1991] son tres: (a) el algoritmo de exploración es más general (para cualquier espacio de búsqueda), (b) su ejecución es más rápida, dado que de una sola vez establece todas las sugerencias y (c) el método se utiliza sobre el GLSR que, al tener mucho menos nodos que un *grid*, permite obtener resultados inmediatos (en un orden de magnitud de milisegundos).

Otra aportación de nuestro trabajo es la *planificación basada en cuadrantes*, que hace posible el desarrollo de un *plan reactivo* en ausencia de modelo. Los planes reactivos no son globalmente óptimos, pero hacen posible la ejecución de misiones en condiciones en las que no puede utilizarse un planificador global y, como ventaja añadida, van completando el

modelo del entorno allí por donde pasan. Este tipo de planificación, por tanto, es especialmente útil en etapas tempranas de funcionamiento del sistema en un nuevo entorno.

La última aportación en lo que respecta al ámbito de la planificación son un conjunto de mecanismos de mantenimiento del modelo —obtenidos a partir del teorema de Bayes— y un sistema de replanificación que entra en acción cuando un subobjetivo se considera inalcanzable.

9.1.3 Navegación reactiva

Nuestras aportaciones sobre navegación reactiva radican en la introducción de algunos refinamientos en la teoría de los campos de potencial artificial estimados en tiempo real a partir de la información suministrada por los sensores del robot.

La primera de ellas —y, a nuestro juicio, la más importante— es la incorporación a la teoría de la capacidad de adición de cargas ficticias como ayuda a la evitación de mínimos locales. Esta técnica constituye un sistema robusto y perfectamente integrado en el contexto del campo de potencial, de manera que no son necesarias "exploraciones por ensayo y error" o "cambios de estrategia" [Latombe, 1993] para sacar al robot de un mínimo local.

En segundo lugar, hemos generalizado las ecuaciones de la fuerza aplicada sobre el robot para cualquier exponente real. El uso de diferentes exponentes permite ajustar el comportamiento del robot para mantenerlo más o menos alejado de los obstáculos, en particular, permite regular cómo influyen en dicho comportamiento los objetos lejanos. Como complemento a esta generalización, hemos deducido un conjunto de ecuaciones que facilitan la obtención de los parámetros necesarios dada la fuerza de repulsión deseada para dos distancias diferentes.

Finalmente, hemos propuesto el método de las cargas ficticias como método general de exploración en espacios de estado. Este nuevo método mejora el rendimiento de la exploración en escalada en lo que a consumo de recursos se refiere, salvando, mediante el uso de cargas ficticias, el problema de los mínimos locales.

9.1.4 Otras aportaciones

En lo que respecta a la arquitectura del sistema, hemos propuesto una arquitectura de robótica móvil *centrada en la memoria*. Esta propuesta responde al papel central que juega la memoria —según los psicólogos cognitivos [Lindsay & Norman, 1983]— en los procesos cognitivos de los seres vivos. Esperamos que un diseño centrado en la memoria sirva como punto de partida a una nueva generación de arquitecturas que supere las limitaciones de las arquitecturas tradicionales.

En otro orden de cosas, durante las pruebas experimentales hemos tenido que enfrentarnos al problema de la calibración del robot. Dado que el robot funciona sobre ruedas, la acumulación de errores de posicionamiento es inevitable. Para contener estos errores hemos utilizado tres procedimientos:

- Establecimiento de un modelo del error y estimación por mínimos cuadrados de los parámetros del error, con objeto de reducir las imprecisiones que se acumulan en cada iteración.
- Recalibración en (x,y) del robot utilizando la información posicional almacenada en los LSRs.
- Recalibración angular utilizando la información sensorial almacenada en los LSRs.

Estas técnicas han permitido el correcto funcionamiento del sistema en condiciones experimentales restringidas. Nuestra sugerencia para un uso industrial es incrementar las capacidades sensoriales del robot, por ejemplo, mediante una brújula digital —que se oferta entre el equipo disponible—, sensores de luminosidad (para recalibrar ayudándose de las fuentes de luz) o visión artificial.

Una última contribución de orden menor ha sido el desarrollo de un *módulo de pilotaje* que utiliza un conjunto sencillo de ecuaciones para el control del robot. Este módulo regula las velocidades de rotación y de translación en función del error angular entre la dirección deseada de movimiento y la dirección actual de las ruedas, y establece la necesaria reducción de velocidad de translación en condiciones de atraque.

Como reflexión final, comentar que el razonamiento espacial humano es un proceso parcialmente inconsciente. Por este motivo es muy difícil plantear modelos que funcionen de modo robusto, ya que tienden a capturar sólo la parte del razonamiento que accede a nuestra consciencia, lo que da lugar a la construcción de modelos que a veces olvidan lo esencial, y que deberán pasar un largo proceso de elaboración hasta llegar a ser modelos válidos. Todos y cada uno de los mecanismos enunciados en esta tesis son producto de un largo proceso de depuración: ninguna idea brillante funciona a la primera, y algunas nunca lo hacen. Sólo hemos aceptado un método cuando éste ha pasado extensas y variadas pruebas experimentales; en el camino han quedado muchas ideas interesantes que nunca llegaron a funcionar.

9.1.5 Nuevas perspectivas

Las propuestas desarrolladas en esta tesis abren, lógicamente, nuevas perspectivas de solución a algunos de los problemas tradicionales de la robótica móvil. Dado el gran número de temas abordados en mayor o menor profundidad por nuestro trabajo, las posibles líneas futuras de investigación y desarrollo son múltiples, distribuidas básicamente en tres direcciones: (a) la profundización en algunos de los problemas planteados, (b) la adaptación de nuestras propuestas a nuevas plataformas y (c) la aplicación de las nuevas técnicas desarrolladas a otros diferentes ámbitos de investigación.

En lo que respecta a la profundización en los problemas de la robótica móvil, las futuras líneas que proponemos son:

- Estudiar la posibilidad de realizar un ajuste automático de ciertos parámetros de la navegación reactiva (intensidad y número de las cargas ficticias, parámetros de repulsión, etc.) para adecuarlos a las condiciones del entorno.
- Mejorar los algoritmos de exploración y planificación basada en cuadrantes, incorporando nuevos heurísticos de decisión en los LSRs.
- Desarrollar en mayor profundidad los modelos de calibración.

En cuanto a la adaptación a nuevas plataformas, podemos citar las siguientes líneas:

- Dotar de mayor autonomía al sistema, incorporando nuevos elementos sensoriales, en particular visión artificial, elementos de control homeostático para recarga de las baterías, etc.
- Desarrollar un interfaz de control para una silla de ruedas, en el cual las sugerencias de navegación serían proporcionadas por el usuario directamente al controlador reactivo. En la misma línea, añadir la posibilidad de definir "etiquetas" en ciertos lugares del entorno, permitiendo la referencia directa a estos lugares por parte del usuario.
- Particularizar los métodos para aplicaciones concretas de carácter industrial o para el sector servicios.

En lo referente a la aplicación de las nuevas técnicas a otros ámbitos, proponemos:

- Desarrollar la idea de *gradiente sensorial* como herramienta general para planificación de tareas y exploración de espacios de estados. Esta es una idea ambiciosa que incluye cuestiones como la determinación de medidas que sustituyan a la información sensorial al trabajar con problemas abstractos.
- Aplicar el algoritmo de búsqueda en oleadas, y por tanto la idea de plan expresado mediante sugerencias, a otros dominios de aplicación, como la planificación de tareas en su sentido más amplio.
- Profundizar en el concepto de arquitectura centrada en la memoria, investigando sus posibilidades en arquitecturas cognitivas para sistemas autónomos en general.

10. Bibliografía

[Abu-Mostafa, 1994] Abu-Mostafa, Y. S. Learning from Hints. *Journal of Complexity*, vol. 10, No. 1. March 1994. pp. 165-178.

[Abu-Mostafa, 1995] Abu-Mostafa, Y. S. Adiestramiento de las Máquinas. *Investigación y Ciencia*. Junio, 1995. pp. 22-27.

[Agre & Chapman, 1990] Agre, R. E.; Chapman, D. What are Plans for? *Designing of Autonomous Agents*. P. Maes (ed.). Cambridge, MA, MIT Press, 1990. pp. 17-34.

[Arkin, 1989] Arkin, R. C. Motor Schema Based Mobile Robot Navigation. *International Journal on Robotics Research*. 8(4), 1989. pp. 92-112.

[Arkin, 1990] Arkin, R. C. The Impact of Cybernetics on The Design of a Mobile Robot System: a Case Study. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 20, No. 6. November/December 1990. pp. 1245-1257.

[Arkin, 1993] Arkin, R. C. Survivable Robotic Systems: Reactive and Homeostatic Control. In Jamshidi, M.; Eicker, P. J. (eds.). *Robotics and Remote Systems for Hazardous Environments*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Arkin, 1994] Arkin, R. C. Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation. *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 3. June 1994. pp. 276-286.

[Azevedo *et al*, 1994] Azevedo, L.; Wann, J. E.; Zato, J. G. *Heart Project E.3.2.- European Curriculum in Rehabilitation Technology Training*. The Swedish Handicap Institute, October, 1994.

[Basye *et al*, 1989] Basye, K.; Dean, T.; Vitter, J. S. Coping With Uncertainly in Map Learning. *Proceedings of The 11th International Conference on Artificial Intelligence*. 1989.

[Booch, 1994] Booch, G. *Object-oriented analysis and design with applications (2ª ed.)*. Benjamin-Cummings Redwood City, California, 1994.

[Brooks & Mataric, 1993] Brooks, R. Mataric, M. Real Robots, Real Learning Problems, in Connell, J. H.; Mahadevan (ed.), S. *Robot Learning*. Kluwer Academic Publishers, Boston, 1993.

[Brooks, 1983] Brooks, R. Solving The Find-Path Problem by Good Representation of Free Space. *IEEE Transactions on System, Man, and Cybernetics*. Vol. SMC 13. No 3. 1983. pp. 190-197.

[Brooks, 1986] Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*. Vol. RA-2, No. 1. March 1986. pp. 14-23.

[Brooks, 1989] Brooks, R. A. The whole iguana. In Brady, M. (ed.). *Robotics Science*. MIT press, Cambridge, MA, 1989.

[Brooks, 1991] Brooks, R. A. Intelligence without Representation. *Artificial Intelligence*. Vol. 47, No. 1-3, Jan 1991. pp. 139-159.

[Brooks, 1992] Brooks, R. A. Artificial Life and Real Robots. In *Toward a Practice of Autonomous Systems. Proceedings of The First European Conference on Artificial Life*. Varela, F. J.; Bourgine, P. (eds.). MIT Press, Cambridge, MA, 1992.

[Card *et al*, 1983] Card, S. K.; Moran, T. P.; Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1983. pp. 23-100.

[Chapanis, 1976] Chapanis, A. Engineering Psychology. In M. D. Dunnette (ed.), *Handbook of Industrial and Organizational Psychology*. Rand McNally, Chicago, 1976.

[Charnley, 1993] Charnley, D. (ed.) *Intelligent Autonomous Vehicles. 1st IFAC International Workshop*. Pergamon Press, Oxford, 1993.

[Chatila, 1982] Chatila, R. Path Planning and Environment Learning in a Mobile System. *Proceedings of The European Conference of Artificial Intelligence*, Orsay, France, 1982.

[Connell & Mahadevan, 1993] Connell, J. H.; Mahadevan (ed.), S. *Robot Learning*. Kluwer Academic Publishers, Boston, 1993.

[Connell, 1990] Connell, J. H. *Minimalist Mobile Robotics, a Colony-style Architecture for an Artificial Creature*. Academic Press, Boston, 1990.

[Conrad, 1951] Conrad, R. Speed and Load Stress in a Sensory-motor Skill. *British Journal of Industrial Medicine*, 1951, 8, pp. 1-7.

[Cox & Wilfong, 1990] Cox, I. J.; Wilfong, G. T. (eds.). *Autonomous Robot Vehicles*. Springer Verlag, New York, 1990.

[Elfes, 1989] Elfes. A. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*. June 1989. pp. 46-57.

[Elfes, 1990] Elfes, A. Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. *Proceedings of The Sixth Conference on Uncertainly in AI*. July 1990.

[Elfes, 1991] Elfes, A. Dynamic Control of Robot Perception Using Stochastic Spatial Models. *Proceedings of The International Workshop on Information Processing in Mobile Robots*. March 1991. pp. 203-218.

[Fisher *et al*, 1991] Fisher, D. H; Pazzani, M. J.; Langley, P. (ed.). *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufman Publishers, San Mateo, 1991.

[Franchi *et al*, 1994] Franchi, P.; Morasso, P.; Vercelli, G. A Hybrid Self-Organizing Architecture for Autonomous Mobile Robots. *ICANN'94 Int. Conf. on Artificial Neural Networks*. Vol. 2. 1994. pp. 1287-1290.

[Freeman, 1991] Freeman, J. A.; Skapura, D. M. *Neural Networks, Algorithms, Applications, and Programming Techniques*. Addison-Wesley, Reading, MA, 1991.

[Gibson, 1977] Gibson, J. J. The Theory of Affordances. In *Perceiving, Acting and Knowing*. R. Shaw and J. Bransford (ed.). Erlbaum, Hillsdale, 1977.

[Halme & Koskinen, 1995] Halme, A. Koskinen, K. (eds.) Intelligent autonomous vehicles: a postprint volume from The 2nd IFAC Conference. Pergamon Press, Oxford, 1995.

[Hertz *et al*, 1991] Hertz, J.; Krogh, A.; Palmer, R. G. *Introduction to The Theory of Neural Computation*. Addison-Wesley, California, 1991.

[Hinton, 1990] Hinton, G. E. Connectionist Learning Procedures, in *Machine Learning Paradigms and Methods*. Ed. Carbonell, J. MIT Press, Cambridge, MA, 1990.

[Hofstadter, 1992] Hofstadter, D. R. *Gödel, Escher, Bach, un Eterno y Grácil Bucle (4ª edición).* Tusquets Editores, Barcelona, 1992.

[Hollier, 1987] Hollier, R. H. (ed.). *Automated Guided Vehicle Systems. International Trends in Manufacturing Technology*. IFS Kempston, Bedford, 1987.

[Humphreys *et al*, 1992] Humphreys, G. W.; Riddoch, M. J.; Boucart, M. The breakdown approach to visual perception: neurophisiological studies of object recognition, in *Understanding Vision*, Humphreys, G. W. (ed.). Blackwell Publishers, Oxford, 1992. pp. 104-125.

[Iyengar & Elfes, 1991a] Iyengar, S. S.; Elfes, A. E (eds.). *Autonomous Mobile Robots, volume 1: Perception, Mapping and Navigation*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[Iyengar & Elfes, 1991b] Iyengar, S. S.; Elfes, A. E. (eds.). *Autonomous Mobile Robots, volume 2: Control Planning and Architecture*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[Kaelbling, 1987] Kaelbling, L. P. An Architecture for Intelligent Reactive Systems. In *Reasoning About Plans and Actions*. Georgeff, P. & Lansky, A. (eds.). Morgan Kauffman, San Mateo, 1987.

[Kanade *et al*, 1994] Kanade, T.; Reed, M. L.; Weiss, L. E. New Technologies and Applications in Robotics. *Communications of The ACM*. Vol. 37. No. 3. March, 1994. pp. 58-67.

[Katib, 1986] Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, Vol. 5, No. 1, Spring 1986. pp. 90-98.

[Keymeulen & Decuyper, 1992] Keymeulen, D.; Decuyper, J. On The Self-organizing Properties of Topological Maps. In *Toward a Practice of Autonomous Systems. Proceedings of The First European Conference on Artificial Life*. Varela, F. J.; Bourgine, P. (eds.). MIT Press, Cambridge, MA, 1992.

[Kohonen, 1982] Kohonen, T. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43. 1982. pp. 59-69.

[Kosaka & Kak, 1992] Kosaka, A.; Kak, A. C. Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties. *CVGIP: Image Understanding*. Vol. 56. No. 3. November, 1992. pp. 271-329.

[Krogh, 1984] Krogh, B. H. A Generalized Potential Field Approach to Obstacle Avoidance Control. In *Proc. Robotics Int. Robotics Res. Conf. Bethelehem*, PA, Aug. 1984.

[Kuipers & Byun, 1988] Kuipers, B.; Byun, Y. A Robust, Qualitative Approach to a Spatial Learning Mobile Robot. *SPIE Vol. 1003 Sensor Fusion: Spatial Reasoning and Scene Interpretation.* 1988. pp. 353-362.

[Kumpel & Serradilla, 1989] Kumpel, D.; Serradilla, F. Global Path Planning Avoiding Unknown Obstacles for an Intelligent Autonomous Robot. *Proceedings of International Workshop on Sensorial Integration for Industrial Robots*. November 1989. pp. 349-351.

[Kumpel & Serradilla, 1990] Kumpel, D.; Serradilla, F. Robot Navigation Systems in a Partially Known Environment Using a Space-Time Learning Graph. *Proceedings of CIM-Europe 6th Annual Conference*. May 1990. pp. 65-75.

[Kurz, 1993] Kurz, A. Building Maps on a Learned Classification of Ultrasonic Range Data. *1st IFAC International Workshop Intelligent Autonomous Vehicles*. Charnley, D. (ed.). Pergamon Press, Oxford, 1993. pp. 193-242.

[Kurz, 1996] Kurz, A. Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*. Vol. 26. No.2. April 1996. pp. 233-242.

[Langton, 1989] Langton, C. G. (ed.). *Artificial Life*. Interdisciplinary Workshop on The Synthesis and Simulation of Living Systems. Addison-Wesley, Redwood City, California, 1989.

[Latombe, 1993] Latombe, J. C. *Robot Motion Planning (3ª ed.)*. Kluwer Academic Publishers, Boston, 1993.

[Leonard & Durrant-Whyte, 1992] Leonard, J. J.; Durrant-Whyte, D. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, 1992.

[Lindsay & Norman, 1983] Lindsay, P. H.; Norman, D. A. *Introducción a la Psicología Cognitiva*. Editorial Tecnos, Madrid, 1983. (Traducción del texto en inglés de 1977).

[Lozano-Pérez & Wesley, 1979] Lozano-Pérez, T.; Wesley, M. A. An Algorithm for Planning Collision-Free Path Among Polyhedral Obstacles. *Communications of The ACM*. Vol. 22, October 1979. pp. 560-570.

[Lozano-Pérez, 1987] Lozano-Pérez, T. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*. Vol. RA-3, No. 3. 1987. pp. 249-265.

[Madarasz *et al*, 1986] Madarasz, R. L.; Heiny, L. C.; Cromp, R. F.; Mazur, N. M. The Design of an Autonomous Vehicle for The Disabled. *IEEE Journal on Robotics and Automation*, Vol. RA-2, No. 3, September 1986, pp. 117-126.

[Maes & Brooks, 1990] Maes, P. Brooks, R. A. Learning to Coordinate Behaviors *AAAI'90*. 1990.

[Maravall *et al*, 1989] Maravall, D.; Mazo, M.; Palencia, V. Robots Móviles. *Revista de la Real Academis de Ciencias Exactas, Físicas y Naturales de Madrid*. Tomo LXXXIII, cuadernos 3º y 4º. Madrid, 1989.

[Maravall, 1996] Maravall, D. Robótica y Robótica Perceptual. Notas de la conferencia impartida en el curso *Neurociencia, Computación y Robots*. El Escorial, 1996.

[Martínez-Arias, 1991] Martínez-Arias, M. R. El Proceso de Toma de Decisiones. En *Tratado de Psicología General 5. Pensamiento e Inteligencia*. Mayor, J.; Pinillos, J. L. (eds.) Alhambra Universidad, Madrid, 1991. pp. 411-494.

[Mataric, 1992] Mataric, M. Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*. Vol. 8. No. 3. June 1992. pp. 304-312.

[Mazo *et al*, 1995] Mazo, M.; Rodríguez, F. J.; Lázaro, J. L.; Ureña, J.; García, J. C.; Santiso, E. Revenga, P.; García, J. Wheelchair for Physically Disabled People with Voice, Ultrasonic and Infrared Sensor Control. *Autonomous Robots*, 2, 1995, pp. 203-224.

[McCormick & Sanders, 1982] McCormick, E. J. Sanders, M. S. *Human Factors in Engineering and Design, Fifth Edition*. McGraw Hill, New York, 1982.

[Meystel, 1991] Meystel, A. *Autonomous Mobile Robots, Vehicles with Cognitive Control*. Word Scientific, Singapore, 1991.

[Mira-Mira, 1996] Mira-Mira, J. Redes Neuronales y Nivel Simbólico. Conferencia impartida en el curso *Neurociencia, Computación y Robots*. El Escorial, 1996.

[Mitchell, 1988] Mitchell, J. S. B. An Algorithmic Approach to Some Problems in Terrain Navigation. *Artificial Intelligence*, 37, 1988. pp. 171-201.

[Neisser, 1976] Neisser, U. *Cognition and Reality: Principles and Implications of Cognitive Psychology*. Freeman & Co., San Francisco, 1976.

[Nilsson, 1969] Nilsson, N. J. A Mobile Automaton: an Application of Artificial Intelligence Tecniques. *Proceedings of The IEEE International Conference on Robotics and Automation*, 1969.

[Nilsson, 1987] Nilsson, N. J. *Principios de Inteligencia Artificial*. Diaz de Santos, Madrid, 1987. (Traducción del texto en inglés de 1980).

[Ó'Dúnlaing & Yap, 1982] Ó'Dúnlaing, C.; Yap, C. K. A Retraction Method for Planning The Motion of a Disc. *Journal of Algorithms*, 6, 1982. pp. 104-111.

[Payton *et al*, 1990] Payton, D. W.; Rosenblatt, J. K.; Keirsey, D. M. Plan Guided Reaction. *IEEE Transaction on Systems, Man, and Cybernetics*. Vol. 20, No 6. November/December 1990. pp. 1370-1382.

[Payton, 1991] Payton, D. Internalized Plans: A Representation for Action Resources. In *Designing Autonomous Agents*. P. Maes (ed.). MIT Press, 1991, pp. 89-103.

[Piaget, 1987] Piaget, J. *Biología y Conocimiento: ensayo sobre las relaciones entre las regulaciones orgánicas y los procesos cognoscitivos (9ª edición)*. Siglo Veintiuno, México, 1987. (Traducción del texto de 1971).

[Pierce & Kuipers, 1994] Pierce, D.; Kuipers, B. Learning to Explore and Build Maps. *Proceedings of The Twelfth National Conference on Artificial Intelligence*. Vol. 2. MIT press, 1994. pp. 1264-1271.

[Pomerleau, 1990] Pomerleau, D. A. Neural Network Based Autonomous Navigation. In *Vision and Navigation: The CMU Navlab*. Kluwer Academic Publishers, Boston, 1990.

[Pomerleau, 1993] Pomerleau, D. A. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston, 1993.

[Rich & Knight, 1994] Rich, E.; Knight, K. *Inteligencia Artificial, 2ª ed*. McGraw Hill, Madrid, 1994. (Traducción del texto en inglés).

[Schoppers, 1987] Schoppers, M. J. Universal Plans for Reactive Robots in Unpredictable Environments. *Proc. Tenth Int. Joint Conf. Artificial Intelligence*. Milan, Italy, Aug. 1987. pp. 1039-1046.

[Serradilla & Kumpel, 1989] Serradilla, F.; Kumpel, D. Robot Navigation in a Partially Known Factory Avoiding Unexpected Obstacles. *Proceedings of Intelligent Autonomous Systems-2*. December 1989. pp. 972-980.

[Serradilla & Maravall, 1996] Serradilla, F.; Maravall, D. A Navigation System for Mobile Robots Using Visual Feedback and Artificial Potential Fields. *Proceedings of The Thirteenth European Meeting on Cybernetics and System Research*. Trappl, R (ed.), April 1996. pp. 1159-1164.

[Serradilla & Maravall, 1997] Serradilla, F.; Maravall, D. A Novel Approach to Autonomous Navigation: The Sensory Gradient Concept and Its Application to Mobile Robots. Enviado a publicación.

[Serradilla, 1992] Serradilla, F. *MOBOT: un Sistema de Planificación de Caminos en Lisp*. Trabajo Fin de Carrera. Facultad de Informática, Universidad Politécnica de Madrid, 1992.

[Shen, 1994] Shen, W. *Autonomous Learning from The Environment*. Computer Science Press, New York, 1994.

[Sigleton, 1971] Singleton, W. T. The Ergonomics of Information Presentation. *Applied Ergonomics*, 1971, 2(4), pp. 213-220.

[Simon, 1969] Simon, H. A. *Sciences of The Artificial*. MIT, Cambridge, MA, 1969.

[Sonka *et al*, 1993] Sonka, M; Hlavac, V; Boyle, R. *Image Processing, Analysis and Machine Vision*. Chapman & Hall, London, 1993.

[Storer & Reif, 1994] Storer, J. A.; Reif, J. H. Shortest paths in the plane with polygonal obstacles. *Journal of the ACM*, 41(5), Sept. 1994. pp. 982-1012.

[Thorpe, 1984] Thorpe, C. E. Path relaxation: path planning for a mobile robot. *Proceedings of The AAAI*, 1984. pp. 318-321.

[Varela & Bourgine, 1992] Varela, F. J.; Bourgine, P. (eds.). *Toward a Practice of Autonomous Systems. Proceedings of The First European Conference on Artificial Life*. MIT Press, Cambridge, MA, 1992.

[Walker *et al*, 1993] Walker, A.; Hallam, J.; Willshaw, D. Bee-havior in a Mobile Robot: The Construction of a Self-Organized Cognitive Map and its Use in Robot Navigation within a Complex, Natural Environment. *ICNN'93 International Conference on Neural Networks*. Vol. 3. 1993. pp. 1451-1456.

[Watt, 1992] Watt, R. J. Visual Analysis and Representation of Spatial Relations. In *Understanding Vision*. G. W. Humphreys (ed.). Blackwell Publishers, Oxford, 1992. pp. 19-38.

[Weiss & Kulikowski, 1991] Weiss, S. M.; Kulikowski, C. A. *Computer Systems That Learn*. Morgan Kaufmann Publishers, San Mateo, 1991.

[Werbos, 1994] Werbos, P. J. *The Roots of Backpropagation, from Ordered Derivatives to Neural Networks and Political Forecasting*. John Wiley & Sons, New York, 1994.

[Winston, 1994] Winston, P. H. *Inteligencia Artificial, 3ª ed*. Addison-Wesley Iberoamericana, Argentina, 1994. (Traducción del texto en inglés).